Grado en Ingeniería Informática

Trabajo de Final de Grado

Proyecto Feedback
Backend y Frontend Web

Autor:
Alex Martínez Martínez

Supervisor:
Sergio Aguado González
Tutor académico:
Angel Pascual del Pobil Ferré

Fecha de lectura: 13 de julio de 2021
Curso académico 2020/2021

Resumen

En este documento se expone la memoria del trabajo de final de grado, en la que se va a detallar el desarrollo de una plataforma de realimentación, donde usuarios expertos en diferentes temas podrán dar consejo a usuarios que quieran aprender. Es una plataforma dirigida a todas las personas que quieran aprender de una forma más dinámica.

El desarrollo de este proyecto consta de, la implementación de la parte del *backend* con *Symfony* y PHP y de la parte del *frontend* web con *Node.js*, concretamente con *ReactJS* y Typescript.

Palabras clave

ReactJS, Symfony, Retroalimentación, Vista web, Servidor

Keywords

ReactJS, Symfony, Feedback, Web Frontend, Backend

Índice general

Índice de cuadros

Índice de figuras

Índice de códigos fuente

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto descrito en este documento se desarrolla en Cuatroochenta[1], que es una empresa tecnológica que está especializada en el desarrollo e implantación de soluciones digitales *cloud*[27] y ciberseguridad con el objetivo de mejorar el rendimiento de las organizaciones. Su principal objetivo es el de contribuir a minimizar la distancia entre lo que sus clientes desean hacer y lo que son capaces de hacer.

La compañía se ha convertido con el paso del tiempo en un socio tecnológico solvente para acometer la transformación digital con las soluciones más potentes de gestión y ciberseguridad.

La idea de realizar este proyecto, surge de la necesidad de que la gente que tiene la intención de aprender sobre diferentes materias sea capaz de recibir retroalimentación de expertos, en su campo de estudio, de una manera mucho más rápida y eficaz. A su vez también busca estandarizar los procesos de aprendizaje, para permitir que muchos usuarios variados sean capaces de acceder a ella sin complicaciones.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto es crear una plataforma que permita que sus usuarios sean capaces de aportar y recibir retroalimentación en cualquier tipo de temática que deseen.

El objetivo principal del proyecto puede llegar a definirse en diferentes objetivos menores:

- Facilitar pedir consejos a expertos que te puedan ayudar a mejorar.
- Facilitar poder dar consejos a aprendices para ayudarles a crecer.
- Ofrecer ayuda a aquellos que deseen aprender.

- Facilitar el acceso a ayudas de aprendizaje.
- Ofrecer una comunicación rápida y sencilla entre aprendiz y mentor.

El sistema debe permitir que aquellos usuarios que quieran aprender, aprendices, sean capaces de realizar publicaciones sobre diferentes temas, y a partir de estas peticiones, aquellos usuarios que tengan conocimiento sobre dichos temas, expertos, puedan instruirlos debidamente a través de publicaciones de retroalimentación. Además de esto, el sistema también debe permitir que los datos de todos los usuarios sean gestionados.

A su vez, el sistema ha de poder gestionar toda la información relativa a las publicaciones, las publicaciones de retroalimentación y las categorías que intervienen en ellas. Con esto se busca permitir que todos los usuarios puedan acceder a esta información de una manera sencilla y eficaz, para propiciar que se adapte a cada usuario según sus necesidades en cada momento.

En cuanto al tema del alcance funcional de este sistema podemos describir las siguientes características. El registro por parte tanto de los aprendices como de los expertos, adicionalmente también existen los usuarios de administración. Todos los usuarios en todo momento van a tener acceso a todas las categorías disponibles, con el objetivo de que los aprendices puedan iniciar el proceso de recibir retroalimentación mientras que los expertos puedan iniciar el proceso de dar retroalimentación.

Las publicaciones de los aprendices se van a dividir según las categorías, para facilitar el acceso a publicaciones concretas se aplicarán filtros a través de dichas categorías. Cualquier experto va a ser capaz de dar retroalimentación a todas las publicaciones. Los expertos también cuentan con la posibilidad de tener categorías favoritas, estas categorías favoritas también van a permitir filtrar las publicaciones.

Las categorías de las peticiones de retroalimentación son creadas por los usuarios de administración del sistema, pero todo usuario, tanto aprendiz como experto, va a poder realizar sugerencias sobre nuevas posibles categorías y ,si estas son adecuadas, se va a permitir que sean añadidas a la lista de categorías.

Cuando los expertos realicen una retroalimentación en una publicación, el aprendiz que haya creado dicha publicación va a poder valorar la respuesta del experto, esas valoraciones se van a incluir en un ranking de clasificación de los expertos mejor valorados. Además de este ranking, vamos a encontrar un ranking de los expertos más activos, los que más retroalimentaciones aportan, y un ranking de las categorías más activas, las que aparecen en mayor número de publicaciones.

Los usuarios, aprendices y expertos, van a ser capaces de consultar su historial y su perfil en todo momento. El sistema debe permitir que todos los usuarios no administradores sean capaces de modificar o eliminar, si así lo desean, sus datos personales.

Además de todo esto, el sistema también va a contar con una herramienta de búsqueda de peticiones concretas, esto, sumado a la posibilidad de filtrar las categorías deseadas, busca facilitar el acceso a las peticiones deseadas. Los usuarios, aprendices, expertos y administradores van a poder marcar peticiones de otros usuarios como indebidas, para evitar conflictos entre usuarios o en la propia aplicación.

Todas las acciones que sean realizadas en el sistema que incluyan, registros, publicaciones, retroalimentaciones, actualizaciones y bajas, van a quedar registradas.

En relación con el tema del alcance informático, se va a emplear *Amazon S3* para almacenar todo el contenido multimedia de las publicaciones y las retroalimentaciones, esto incluye documentos, vídeos e imágenes.

También se va a utilizar una base de datos *MySQL* que va a alojarse en la nube y que va a almacenar toda la información necesaria del sistema, publicaciones, retroalimentaciones, categorías, entre otras.

Para finalizar, con respecto al alcance organizativo del proyecto, este implica al departamento de desarrollo web, el equipo de *UX/UI* (diseño de interfaces) y el departamento de desarrollo de aplicaciones móviles, ya que se va a trabajar conjuntamente.

1.3. Descripción del proyecto

Como se ha mencionado en la sección anterior, este proyecto se va a realizar en conjunto entre el departamento de desarrollo web y el departamento de desarrollo móvil. Esto implica que va a haber dos grupos de trabajo, el departamento de desarrollo web se va a encargar de desarrollar el *backend* del sistema y un *frontend* web, mientras que el departamento de desarrollo móvil se va a encargar de desarrollar una aplicación móvil.

Yo me encuentro en el departamento de desarrollo web, por lo que mi rol en este proyecto es el de desarrollar el *backend* y el *frontend* web.

Actualmente en el mercado hay muchas soluciones que tratan de resolver el mismo problema que nosotros queremos resolver, que es el de permitir que la gente aprenda sobre nuevos temas, y que para ello reciba ayuda de expertos en dichos temas. Todas las soluciones que existen abarcan como resolver este problema de diferentes maneras.

Nosotros, con este proyecto, buscamos crear una versión completamente diferente al resto, no nos vamos a dedicar a mejorar las soluciones ya creadas. Debido a esto, el desarrollo de este proyecto comenzará desde cero, no incluiremos conceptos de otras plataformas, vamos a intentar conseguir que nuestra plataforma sea una invención completamente nuestra, que intente solucionar de una manera única el problema planteado.

Para lograr esto, no nos vamos a inspirar en conceptos empleados previamente en proyectos que resuelvan este problema o alguno similar. Todas las ideas que nos surjan se intentarán desarrollar al máximo nivel posible, para intentar alcanzar un alto nivel de diferenciación y crear una solución que permita, por sí sola, superar al resto de soluciones que ya existen.

1.3.1. Tecnologías y herramientas

A continuación, se van a exponer todas las tecnologías y herramientas que se han requerido durante el desarrollo del proyecto.

Para gestionar adecuadamente la planificación y el seguimiento del proyecto, se emplea la herramienta *Jira Software*, esta es la herramienta que 480 usa en todos sus proyectos, por lo que es obligatoria su utilización. Dentro de esta herramienta, nuestro papel es el de emplear una pizarra ágil que permite recoger adecuadamente todos los datos relevantes al proyecto, que se expondrán en detalle en el capítulo 2. Una vez estos datos se expongan y se definan, esta herramienta ayuda en mantener información sobre el estado del proyecto.

En cuanto al control de versiones[1], se va a emplear *Git*[9] y *Github*[2], esto nos va a permitir guardar un estado sobre el desarrollo actual del proyecto. Este control de versiones en la nube, nos va a dar la opción de automatizar sus despliegues. También va a facilitar el acceso al código por parte del *Scrum Master* y del resto de los equipos de desarrollo.

Hay que mencionar que para el desarrollo del *backend* y del *frontend* se emplean diferentes herramientas y tecnologías.

Backend

El desarrollo del *backend* incluye el desarrollo de los *webservices*[3] y lógica de negocio con el *framework*[4] *Symfony*, que emplea el lenguaje de programación PHP[3] (del inglés, *Hypertext Pre-Processor*). El *backend* va a desarrollarse como una *API REST*[13], por lo que se va a emplear la especificación OpenAPI[17].

Este desarrollo se va a llevar a cabo en el IDE (Entorno de Desarrollo Integrado) *PHPStorm*, perteneciente a la empresa *JetBrains*.

Frontend web

El desarrollo del *frontend* web incluye el desarrollo de la interfaz de usuario y de la comunicación con el *backend*. Para el desarrollo del *frontend* web se va a hacer uso de *Node.js*[10] de *ReactJS*[8], se va a hacer uso del lenguaje de programación Typescript[16], aunque originalmente estaba planteado hacer uso de Javascript[19]. En cuanto al tema del diseño de la interfaz de usuario, se va a usar *Bootstrap*[6], para que la aplicación cumpla unos estándares de diseño adecuados.

Este desarrollo se va a llevar a cabo en el IDE (Entorno de Desarrollo Integrado) *Visual Studio Code*, perteneciente a la empresa *Microsoft*.

[1] Gestión de todos los cambios que se realizan en un proyecto de software
[2] Plataforma para alojar un proyecto en la nube, empleando Git
[3] Protocolos y estándares utilizados para intercambiar datos entre diferentes aplicaciones
[4] Entorno de trabajo con estructura preestablecida para desarrollar *software*

1.4. Estructura de la memoria

Esta memoria va a exponer el desarrollo del proyecto, esta explicación sobre el proyecto se va a dividir en diversos capítulos, a continuación voy a exponer los contenidos que se presentarán en cada capítulo.

En el capítulo 2 se detalla la planificación del proyecto, esto incluirá una explicación de la metodología que se va a emplear, un estudio sobre los costes del proyecto e informes de seguimiento de los Sprints del proyecto.

En el capítulo 3 se expone el análisis detallado del sistema, se van a incluir los casos de uso que lo conforman, además también se reflejan los diseños de la arquitectura y la base de datos. Además de esto se especifica la guía de estilos que se ha empleado.

En el capítulo 4 se detalla el desarrollo del proyecto, se va a explicar como este se ha llevado a cabo, que problemas han surgido y los resultados del mismo. También se detallan las pruebas que se han llevado a cabo sobre el proyecto.

En el capítulo 5 se refleja mi opinión sobre todo lo acontecido en la estancia en prácticas y sobre el desarrollo del proyecto.

Capítulo 2

Planificación del proyecto

2.1. Metodología

La metodología que se ha empleado para el desarrollo de este proyecto es la metodología *Scrum*[2], esta se encuentra dentro del conjunto de las metodologías ágiles[26]. Se ha optado por esta metodología debido a dos razones, la primera es que es la habitual forma de trabajo de la empresa y la otra es porque se trata de la metodología apropiada para este tipo de proyectos. Este proyecto presenta un rango de incertidumbre a tener en cuenta, en cuanto a los requisitos funcionales se refiere, dado que como se va a poder observar a lo largo de este capítulo, algunos de ellos tuvieron que adaptarse a las nuevas peticiones y/o ampliarse.

Para hacer uso de la metodología *Scrum* el primer paso a realizar es el de crear la pila del producto (*product backlog*), en la cual se incluyen todas las historias de usuario[15] y tareas a realizar en el proyecto. Una vez se han registrado todas las tareas y las historias en la pila del producto, estas se ordenan y se estiman, todo esto con el propósito de obtener aquellas que sean más prioritarias.

Tras priorizar las historias, se organizan los Sprints, estos Sprints se ejecutarán uno detrás de otro de manera secuencial. Estos Sprints siempre tendrán la misma duración, en el caso de este proyecto esta duración será de dos semanas. Durante la realización de todos los Sprints se realizarán diferentes reuniones cuyo objetivo es el establecer los objetivos de dicho Sprint, darles un orden según su importancia y estipular su desarrollo. Además de estas reuniones también se realizan reuniones antes y después de los Sprints, con los mismos objetivos.

A continuación se muestran las reuniones que se llevan a cabo, como estas son realizadas y los roles que intervienen en cada una de ellas:

- Reunión de planificación (*Sprint planning*): Reuniones realizadas antes de comenzar cada Sprint. A estas reuniones debían asistir el equipo de desarrollo web, el equipo de desarrollo móvil, el *Scrum Manager* y el *Product Owner*.
- Reunión diaria (*Daily meeting*): Reuniones realizadas durante todos los días durante los

cuales el Sprint se encontraba en ejecución. A estas reuniones debíamos asistir tanto el equipo de desarrollo web como el equipo de desarrollo móvil para poder coordinar nuestro trabajo adecuadamente. Con esto se buscaba que ninguna de las partes quedara estancanda en el desarrollo de alguna funcionalidad.

- Reunión de cierre (*Sprint Review*): Reuniones realizadas al acabar formalmente cada Sprint. A estas reuniones debían asistir el equipo de desarrollo web, el equipo de desarrollo móvil, el *Scrum Manager* y el *Product Owner*. Durante la ejecución de estas reuniones los diferentes equipos de desarrollo mostraban a su supervisor, que se hacia pasar por el *Product Owner*, todos sus progresos durante el Sprint.

A continuación se describen los roles mencionados anteriormente, además se van a incluir las responsabilidades que deben llevar a cabo para que la metodología *Scrum* funcione adecuadamente.

- *Product Owner*: Este rol fue llevado a cabo por el supervisor asignado de la empresa, dado que se trataba de un proyecto interno y no presentaba uno real. Durante la ejecución de las reuniones que se llevaron a cabo con los dos equipos de desarrollo, nos indicaba las principales funcionalidades en las que deberías enfocarnos principalmente a la hora de implementar el producto. Además de esto, también se encargaba de revisar el trabajo realizado hasta ese momento, para comunicarnos si era correcto o deberíamos realizar alguna mejora.
- *Scrum Master*: Este rol fue llevado a cabo por los dos equipos de desarrollo, pero en cada Sprint cambiaba quien era el que llevaba a cabo el rol, en algunos Sprints era yo mientras que en otros era la persona encargada del *frontend* móvil. Decidimos llevarlo a cabo de esta manera debido a que había pocas personas involucradas en realizar el proyecto y nos aportaría experiencia en este tipo de roles y además evitaríamos que se pudiesen generar egos entre los diferentes equipos.
- Equipo de desarrollo: En este proyecto encontramos dos equipos de desarrollo, aunque cada uno esta únicamente formado por una persona. El primer equipo es el de desarrollo web, en el que me incluyo, mientras que el segundo equipo es el de desarrollo móvil, en el que se incluye el compañero que se encargó de realizar la aplicación móvil.

2.2. Planificación

En cuanto a la metodología empleada en este proyecto, a lo largo de esta sección se va a exponer la planificación del proyecto, en la que se incluye la pila del producto y la estimación de las historias que se encuentran dentro de ella.

Antes de comenzar a desarrollar el producto, con el objetivo de obtener conocimiento de las tecnologías que se emplearían en él y para ser capaces de estimar con mayor precisión la dificultad de llevar a cabo todas las historias de usuario, se realizó una formación previa que ocupó dos semanas de trabajo. En mi caso esta formación fue un poco diferente ya que la realicé tanto para PHP como para *ReactJS*.

Originalmente solo me dediqué a completar la formación en PHP, ya que el supervisor estipuló que era lo que necesitaba para poder realizar la estimación y que primero debía centrarme en el *backend* plenamente, antes de poder comenzar con el *frontend* web. Debido a esto completé la formación de la parte del *backend*, tras esto realizamos la planificación, desarrollé el servidor, y comencé con la formación del *frontend* para después comenzar su desarrollo.

Una vez finalizada la formación inicial, se continuó con la especificación de requisitos y se estimaron las historias de usuario que formaban la pila del producto. En cuanto a esto, se determinó que la pila sería compartida tanto por *backend* como por el *frontend*. Las historias de usuario fueron definidas en una reunión con el *Product Owner*, en la que nos indicó todas las funcionalidades con las que debería contar el producto, y además de eso, también nos indicó cuáles de ellas eran más prioritarias y debíamos realizar lo antes posible.

Cada historia de usuario se estimó con puntos de historia[22], para estipular cuantos puntos de historia se le otorgaban a cada una, primero se adecuó cual de todas las historias era más compleja y se le propició una puntuación acorde a dicha complejidad. Una vez esa historia se estimó, el resto de historias fueron estimadas en función de su complejidad relativa y la posible experiencia previa en ellas.

Una vez se llevó a cabo la estimación, la pila del producto fue ordenada de acuerdo a las peticiones del *Product Owner*, para que esas historias acabaran en el primer Sprint. Tras esto, el resto de historias fueron ordenadas, de acuerdo con los criterios tanto del *Scrum Master* como del equipo de desarrollo, esto implicó que se priorizaran aquellas historias que podrían aportar más al usuario final. En el cuadro 2.1 se puede observar la pila del producto después de realizar todos los pasos expuestos previamente.

Identificador	**Descripción**	**Estimación (puntos de historia)**
TA01	*Creación de la base de datos.*	-
TA02	*Diseño interfaces.*	-
HU01	*Como aprendiz.* *Quiero poder pedir feedback de los expertos utilizando categoría, título y descripción.* *Para poder mejorar en esa categoría.*	7ptos
HU02	*Como experto.* *Quiero poder tener un listado con las peticiones de feedback.* *Para poder ayudar a los usuarios que lo necesiten.*	4ptos
HU03	*Como administrador.* *Quiero poder crear categorías y subcategorías.* *Para que los usuarios y expertos puedan recibir y dar feedback.*	2ptos
HU04	*Como experto.* *Quiero poder marcar las categorías como favoritas.* *Para poder consultar con mayor celeridad las categorías que más me gustan o a las cuales soy más afín.*	3ptos
HU05	*Como experto.* *Quiero poder dar feedback a publicaciones de diferentes categorías.* *Para poder ayudar a los usuarios que necesitan ser retroalimentados.*	7ptos
HU06	*Como usuario.* *Quiero poder consultar mi historial sobre las publicaciones realizadas.* *Para poder consultarlas en un momento dado.*	4ptos
HU07	*Como usuario.* *Quiero poder consultar mis datos.* *Para poder visualizarlos en un momento dado.*	4ptos
HU08	*Como aprendiz.* *Quiero poder ver todas las categorías.* *Para poder iniciar el proceso de pedir feedback.*	2ptos
HU09	*Como experto.* *Quiero poder ver todas las categorías.* *Para poder iniciar el proceso de dar feedback.*	2ptos
HU15	*Como usuario.* *Quiero poder marcar una petición como indebida.* *Para evitar conflictos entre usuarios o en la propia aplicación.*	3ptos
HU16	*Como usuario.* *Quiero poder buscar peticiones concretas.* *Para facilitar el acceso a dicha petición.*	3ptos

Identificador	**Descripción**	**Estimación (puntos de historia)**
HU11	*Como aprendiz.* *Quiero poder dar una puntuación sobre el feedback obtenido.* *Para dar una mejor reputación a los expertos que lo merezcan*	3ptos
HU12	*Como usuario.* *Quiero poder observar un ranking de clasificación de los expertos mejor valorados.* *Para poder comprobar su reputación en la aplicación.*	4ptos
HU13	*Como usuario.* *Quiero poder observar un ranking de las categorías más activas* *Para poder tener en cuenta el tiempo aproximado en el que recibiré ayuda.*	4ptos
HU14	*Como usuario.* *Quiero poder observar un ranking de los expertos más activos.* *Para poder tener en cuenta el tiempo aproximado en el que recibiré ayuda.*	4ptos
HU10	*Como usuario.* *Quiero poder hacer sugerencias mediante un buzón.* *Para solicitar nuevas categorías.*	2ptos
HU17	*Como usuario.* *Quiero poder registrarme en la aplicación.* *Para poder usarla.*	4ptos
HU18	*Como usuario* *Quiero poder modificar mis datos.* *Para poder adaptarlos a las necesidades del momento.*	3ptos
HU19	*Como usuario* *Quiero poder borrar mis datos.* *Para poder dejar de formar parte de la aplicación.*	2ptos

Cuadro 2.1: Product backlog con las historias priorizadas y estimadas.

2.3. Gestión de riesgos

En esta sección se va a exponer otro de los puntos claves para la correcta planificación de un proyecto software, la gestión de riesgos. El objetivo de esta gestión es el de identificar, prevenir y corregir, los riesgos que pueden surgir durante el desarrollo de un proyecto. Esta gestión es de suma importancia, ya que ni no se lleva a cabo, es posible que el proyecto no pueda completarse de una manera satisfactoria.

Los riesgos representan la posibilidad de que se produzca un problema, de mayor o menor importancia, durante el desarrollo del proyecto. La aparición de estos riesgos puede no llegar a darse durante el desarrollo del proyecto, pero a pesar hay que identificarlos adecuadamente. Este análisis de los riesgos incluye tres fases:

- Descripción del riesgo: En esta fase simplemente damos un identificador y un nombre al riesgo que puede aparecer. Además de esto, indicamos el tipo de riesgo, que indica en que momento puede llegar a aparecer, ya sea en el proyecto, en el producto o en ambos. El resultado de esta fase puede se puede observar en el cuadro 2.2.
- Análisis del riesgo: En esta fase se describen en detalle todos los riesgos que se han identificado en la fase anterior. Para el correcto análisis de un riesgo, debemos de tener en cuenta, la magnitud, es decir, el nivel de dificultad que puede tener, la descripción detalla del riesgo, el impacto, es decir, en que afectará al proyecto o producto y finalmente los indicadores, es decir, los elementos que nos hacen ver que el riesgo se esta produciendo. El resultado de esta fase puede se puede observar en el cuadro 2.3.
- Planes de prevención y contingencia: En esta fase se definen los planes de prevención y de contingencia para cada riesgo. Los planes de prevención hacen referencia a las pautas que se deben de seguir para que el riesgo no llegue a producirse. Los planes de contingencia exponen que hacer en caso de que el riesgo llegue a ocurrir. El resultado de esta fase puede se puede observar en el cuadro 2.4.

Como se verá más en detalle en la sección 2.5, algunos de estos riesgos llegan a surgir, aunque no en todo su potencial, provocando que tengamos que enfocarnos en solucionar los problemas que nos surgen y que no podamos llevar a cabo el desarrollo como se había planeado. Estos riesgos son a nivel de proyecto y producto, por lo que este análisis se ha hecho en conjunto con ambos equipos de desarrollo y ambos equipos han de ser conscientes de ellos.

ID	Descripción del riesgo	Tipo de riesgo
R01	Falta de experiencia con las herramientas utilizadas	Riesgo del producto
R02	Abandono temporal de un miembro del equipo	Riesgo del proyecto
R03	Falta de comunicación entre los integrantes del equipo.	Riesgo del proyecto
R04	Diseño erróneo	Riesgo del producto
R05	Restricciones tecnológicas	Riesgo del producto
R06	Falta de tiempo o inexperiencia al gestionarlo.	Riesgo del proyecto
R07	Inestabilidad en el lugar de desarrollo.	Riesgo del proyecto

Cuadro 2.2: Listado de riesgos.

ID	Análisis del Riesgo
R01	*Magnitud*: Media durante todo el desarrollo y en cada uno de los Sprints. *Descripción*: El equipo no tiene experiencia previa utilizando las herramientas con las que se desarrolla el producto, lo cual puede suponer una dificultad a la hora de alcanzar los objetivos establecidos. *Impacto*: Retraso en la entrega del proyecto y/o coste adicional. *Indicadores*: El desarrollo real del proyecto no se corresponde con la planificación realizada.
R02	*Magnitud*: Alta, cuando afecta a un solo miembro. Muy alta, si afecta a más de uno. *Descripción*: Algún miembro del proyecto no se encuentra disponible por cualquier problema. Dada la situación actual, la baja por enfermedad puede darse de manera más habitual. *Impacto*: La falta de disponibilidad de los recursos humanos puede provocar retrasos en la finalización completa de los diferentes Sprints y en las historias en las que se encuentran involucrados. Teniendo en cuenta que la entrega no puede posponerse, la falta de disponibilidad de personal puede suponer una pérdida de calidad en el producto o un incremento en el coste. *Indicadores*: No procede.
R03	*Magnitud*: Media. *Descripción*: Aparición de problemas y discrepancias entre los miembros del proyecto. Falta de acuerdo en las decisiones tomadas. *Impacto*: Si los desacuerdos no son rápidamente resueltos se pueden provocar retrasos en la ejecución de los Sprints. Estos retrasos podrían suponer que el proyecto no se completara y que el trabajo de los Sprints no fuera completado. *Indicadores*: Mucho tiempo dedicado a decisiones concretas, énfasis en las posturas enfrentadas, número de enfrentamientos con respecto a una misma decisión.
R04	*Magnitud*: Alta, sobretodo en los Sprints finales *Descripción*: La interfaz de usuario no satisface los deseos del Product Owner o los usuarios no son capaces de interactuar adecuadamente con ella, o la describen como una interfaz muy complicada o extraña. *Impacto*: Puede suponer retrasos ya que se debería de rehacer la interfaz de usuario. *Indicadores*: Los resultados de las pruebas de usuario no son satisfactorios. El *Product Owner* nos comunica que no está de acuerdo con la interfaz de usuario.

ID	Análisis del Riesgo
R05	*Magnitud*: Media. *Descripción*: Debido a las restricciones a la hora de utilizar ciertas herramientas en el proyecto, se puede ver limitada la capacidad de flexibilizarlas y utilizar otras tecnologías que faciliten la realización de ciertas partes del proyecto. *Impacto*: Puede suponer retrasos a la hora de terminar el Sprint o si se encuentra en las últimas fases del proyecto puede existir un retraso temporal que incluya un coste adicional. *Indicadores*: Complicaciones especiales a la hora de implementar una característica del sistema.
R06	*Magnitud*: Alta sobretodo en los Sprints finales del proyecto *Descripción*: Saber gestionar el tiempo es clave a la hora de realizar un proyecto ya que en caso de que se haga una mala gestión o que no haya tiempo suficiente puede suponer una gran cantidad de errores que se dejan sin cubrir o incluso la imposibilidad de poder finalizar el producto *Impacto*: El tiempo es oro, y por eso mismo, ser incapaces de ajustarse al tiempo puede suponer grandes retrasos, que impliquen una monetización adicional que no se puede obtener. *Indicadores*: La resolución de las tareas no se ajusta a la gestión del tiempo establecida.
R07	*Magnitud*: Media durante todo el desarrollo. *Descripción*: El grupo tiene dificultades para poder trabajar en su lugar de trabajo. *Impacto*: La imposibilidad de trabajar en un entorno adecuado, puede suponer grandes retrasos. *Indicadores*: No procede.

Cuadro 2.3: Descripción de riesgos.

ID	Plan de prevención/acción	Plan corrección/contingencia
R01	Realizar cursos de formación sobre las herramientas que se van a utilizar a todos los miembros del equipo.	Contactar con los expertos en la materia de la empresa para que ayuden a los miembros que están teniendo problemas.

ID	Plan de prevención/acción	Plan corrección/contingencia
R02	Tener un margen temporal en cada sprint del proyecto para que la pausa temporal del trabajo no afecte al desarrollo del proyecto.	Buscar otras formas de trabajo para que el desarrollo no se detenga por dicho problema. Ej: Teletrabajo. En caso de que no fuera posible, se trataría de realizar tareas que no necesiten al otro miembro del equipo.
R03	Cada vez que se fije un punto de dirección en el proyecto, todo tiene que quedar totalmente claro, sin dudas y con la aceptación total de todos los miembros del grupo.	Se establecen las siguientes reglas para definir una política de toma de decisiones en caso de desacuerdo. Las cuestiones relativas a las historias de usuario se tratarán junto al Product Owner, que será quien tome la decisión. Las cuestiones de diseño o técnicas se tratarán en conjunto entre los dos equipos de desarrollo, donde cada uno aportará su opinión.
R04	Tratar de consensuar un diseño con el Product Owner y realizar múltiples pruebas de usuario cada ciertas iteraciones como forma de prevención.	Reunión entre todos los miembros de desarrollo, incluido el Product Owner, para consensuar las correcciones necesarias en la interfaz.
R05	Asegurarse de que el equipo se familiariza con las herramientas y corrobora su potencial antes de empezar a trabajar.	Contactar con los expertos en la materia de la empresa para que ayuden a los miembros que están teniendo problemas.
R06	Realizar un estudio sobre proyectos de calibre similar para poder realizar una gestión del tiempo más precisa.	Redefinir los tiempos conforme se va desarrollando el proyecto, para poder ser capaces de dar información más precisa sobre el tiempo restante.
R07	La inestabilidad del lugar de trabajo es imprevisible.	Medir las posibles alternativas a realizar, se podría buscar un nuevo lugar de trabajo que no suponga grandes costos adicionales. También cabe la idea de usar las funciones de teletrabajo, si los trabajadores tienen la confianza de poder trabajar adecuadamente en el entorno del hogar.

Cuadro 2.4: Acciones de prevención y corrección de los riesgos.

2.4. Estimación de recursos y costes del proyecto

En este sección se va a exponer la estimación de los costes del proyecto, esta estimación involucra la totalidad del proyecto, es decir, se van a tener en cuenta todas las partes implicadas en él. Por esto mismo se evaluarán tanto el departamento web, *backend* y *frontend* web, como el departamento móvil, *frontend* móvil.

Lo primero que hay que tener en cuenta para realizar la estimación es el coste proporcional de los activos *Hardware* con los que se trabajó en el proyecto. El coste proporcional referencia el coste de usar dicho activo durante las horas que se ha requerido, también se debe tener en cuenta en cuanto tiempo se puede amortizar su compra.

Para ejemplificar esto, usemos el activo Ordenador sobremesa del *backend*. El coste total (CT) del activo fue 2.700,00 €, ese activo tiene un tiempo de amortización (TA) de 6 años, que equivalen a 72 meses (este tiempo deberá ser evaluado en horas), y se ocupó (TU) durante 100 horas del desarrollo. Esto equivaldría a la ecuación 2.1. El resultado de aplicar esto se puede observar en el cuadro 2.5

$$CP = (CT \div TA \div 30 \div 24) \times TU \tag{2.1}$$

Proyecto	**Hardware**	**Coste total (€)**	**Coste proporcional (€)**
Backend	Ordenador sobremesa	2.700,00	5,21
Frontend móvil	Ordenador sobremesa	1.800,00	10,42
	Móvil One Plus 8 Pro	700,00	12,15
Frontend web	Ordenador sobremesa	2.700,00	10,42
Total		7.900,00	38,19

Cuadro 2.5: Costes Hardware del proyecto.

El siguiente elemento a tener en cuenta en la estimación de costes es el coste de todas las herramientas empleadas en el desarrollo del proyecto. Estas herramientas hacen referencia a todo software que ha ayudado de cualquier modo al proyecto, ya sea durante su planificación, su análisis, su implementación o su verificación. Lo que nos interesa a tener en cuenta sobre este coste, se trata del coste total durante el tiempo de desarrollo, al contrario que el Hardware. El resultado de calcular los costes relativos a los activos software se puede observar en el cuadro 2.6.

Proyecto	**Software**	**Coste total (€)**
Común	Jira Software	0,00
Backend	PHPStorm	0,00
	AWS	1,00
	Postman	0,00
	Heroku	0,00
Frontend móvil	Android Studio	0,00
	Balsamiq Frameworks Free Trial	0,00
Frontend web	Visual Studio Code	0,00
	Heroku	0,00
Total		1,00

Cuadro 2.6: Costes Software del proyecto.

Una vez los costes en hardware y software han sido evaluados, el siguiente componente a tratar es el de los sueldos de los trabajadores. Como puede observarse en el cuadro 2.7 el sueldo proporcional de un empleado, concretamente un programador *junior* es de 1.153,85 €.

Este sueldo proporcional (SP) hace referencia al sueldo por hora (SH), es importante mencionar que para este proyecto se va a trabajar a media jornada durante todas las horas (H) del desarrollo. Se va a partir de un sueldo anual (SA) base de 16.000,00 €. El resultado de aplicar esto se puede observar en el cuadro 2.7.

$$SH = SA \div 52 \div 40 \tag{2.2}$$
$$SP = (SH \div 2) \times H \tag{2.3}$$

Sueldo Anual (€)	**Sueldo/hora(€)**	**Sueldo proporcional (€)**
16.000,00	7,69	1.153,85

Cuadro 2.7: Costes en sueldos del proyecto.

Finalmente, tras evaluar el coste en sueldos, podemos obtener el coste total (CT) del proyecto. Este coste viene dado, por el coste hardware (CH), el coste software (CW), el coste en sueldos (CS), también se debe tener en cuenta que en el proyecto intervienen dos desarrolladores, este coste se doblará, y finalmente aplicar costes indirectos (CI).

Estos costes indirectos hacen referencia a todos los costes que propician que el lugar de trabajo sea adecuado. Habitualmente este coste supone entorno a un 20 % extra sobre el coste total del proyecto sin ese sobrecoste (CTS).

$$CTS = CH + CW + CS \times 2 \tag{2.4}$$
$$CI = CTS \times 0,2 \tag{2.5}$$
$$CT = CTS + CI \tag{2.6}$$

Si se aplican las formulas anteriores el coste total del proyecto resultaría en 2.816,26 €.

$$CTS = 38,19 + 1 + 1{,}153,85 \times 2 \tag{2.7}$$
$$CI = 2{,}346,89 \times 0,2 \tag{2.8}$$
$$CT = 2{,}346,89 + 469,38 \tag{2.9}$$

Si se evalúa detalladamente el coste total final del proyecto, este resulta en 9,39 €/hora. Este coste por hora es totalmente asumible para un proyecto de este calibre y con estas características, que ha implicado únicamente a dos programadores *junior* que están comienzan su labor de trabajadores. Según los conocimientos previos aportados por la empresa si el coste de un proyecto como este no supera los 30 €/hora es un proyecto factible y del que se va a poder obtener un beneficio adecuado.

2.5. Seguimiento del proyecto

En este sección se exponen todos los detalles sobre el seguimiento del proyecto, estos detalles incluyen todos los Sprints desarrollados en el proyecto, por cada uno de estos Sprints se va a exponer la reunión de planificación, la de cierre, y algunas observaciones sobre dicho Sprint.

Cabe mencionar que este seguimiento es conjunto para el departamento web y el departamento móvil, por eso mismo todas las figuras y cuadros que aparecen en esta sección involucran a ambas partes. Si en algún momento hay alguna disparidad entre lo expuesto y las figuras o cuadros, es debido a que no toman en cuenta únicamente mi aportación al proyecto, sino la de todas las partes implicadas.

2.5.1. Sprint 1

Reunión de planificación

En la reunión de planificación del Sprint 1 se definió el objetivo el mismo. Este objetivo consistía en la creación tanto de categorías como de publicaciones, por parte de los administradores y aprendices respectivamente, además estas últimas deberían de poder obtener retroalimentaciones por parte de los expertos. También incluía que se pudieran marcar categorías como favoritas.

Tras haber definido los objetivos del Sprint se pudo comenzar con el desarrollo del mismo, para esto se siguió el orden de priorización de las tareas e historias de usuario.

Observaciones

El desarrollo de este Sprint comenzó con la parte del *backend* en conjunto con la parte del *frontend* móvil. En este inicio conjunto se estipuló el diseño de la base de datos, en el cual se incluirían todas las entidades que deberían formar parte de ella y las relaciones entre ellas.

Una vez que esto se llevó a cabo mi siguiente tarea fue la de organizar el esqueleto del proyecto para documentar la API REST, esto implicó generar un rutado básico sobre el que poder trabajar y expandir con las diferentes funcionalidades. Tras esto organicé un entorno de despliegue automático en Heroku, para que el *frontend* móvil pudiera probar todas las funcionalidades que se añadirían posteriormente.

Une vez todo eso estuvo terminado, comencé a desarrollar las funcionalidades que había que implementar en este Sprint. Estas fueron completados sin mayores complicaciones, debido a esto como aún quedaba mucho tiempo del Sprint, comencé a desarrollar las funcionalidades que faltaban en la pila del producto por orden de priorización.

Durante la ejecución de este Sprint se me comunicó, a mi exclusivamente debido a que avanzaba rápidamente, que en un futuro deberíamos añadir nuevas funcionalidades, es decir,

nuevas historias a la pila del producto. Se me adelantó una de ellas, que se trataba de la funcionalidad de autenticarse en la aplicación.

Debido a esto, me dediqué a realizar un sistema de autenticación, se decidió usar una identificación mediante JWT (del inglés, *JSON Web Token*). Esta autenticación supuso un problema, ya que la documentación sobre como hacer uso de ella era prácticamente inexistente y sus posibilidades eran sumamente altas. Estuve un poco atascado en esta parte, pero finalmente fui capaz de completarlo.

Reunión de cierre

En la reunión de cierre se realizó una demostración del trabajo desarrollado hasta la fecha, esta demostración se hizo en conjunto entre los dos departamentos de desarrollo y el *Product Owner*.

Esta demostración fue satisfactoria por parte del *backend*, debido a que el avance de este había sido bastante amplio, por lo que el Sprint se consideró acabado por mi parte.

En cambio el *frontend* móvil no fue capaz de acabar todas las tareas asignadas al Sprint 1, es debido a esto que en la figura 2.1 el Sprint 1 no es representado como finalizado.

Debido a haber completado todas las tareas del Sprint satisfactoriamente y de completar tareas adicionales que se incluirían en futuros Sprints del desarrolló, se me hizo saber que en cuanto acabará el desarrollo de las funcionalidades restantes debería comenzar a desarrollar el *frontend* web no sin antes realizar la formación sobre el mismo.

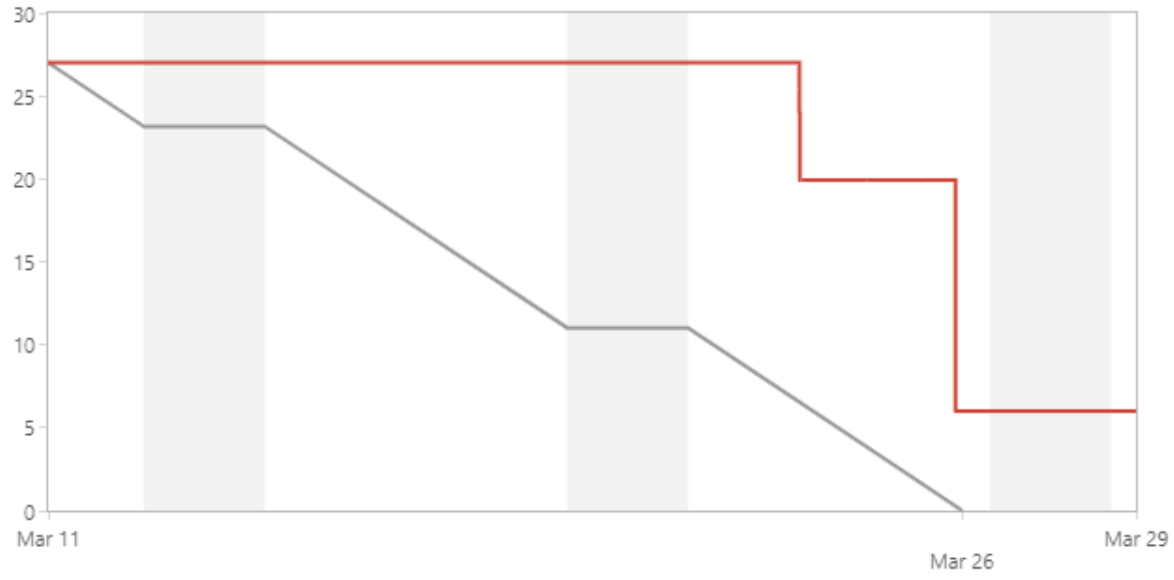

Figura 2.1: Gráfico *Burn Down* del Sprint 1.

2.5.2. Sprint 2

Reunión de planificación

En la reunión de planificación del Sprint 2 se definió el objetivo del mismo, este incluía que los usuarios pudieran observar sus datos y sus acciones en la aplicación, y que además pudieran identificarse como usuarios de la misma y acceder a ella.

Como ya se había adelantado en el Sprint anterior, este objetivo provocó que se tuvieran que añadir dos nuevas historias a la pila del producto, estas historias quedan reflejadas en el cuadro 2.8.

Debido a esto mi trabajo a realizar en este Sprint era el de completar primero la nueva historia de usuario HU20, para posteriormente acabar con el resto de historias inacabadas de la pila del producto.

Cabe mencionar que debido a que había un periodo de vacaciones, durante la ejecución de este Sprint, su duración sería menor a las dos semanas, por lo que esta se extendió 4 días para que la duración fuera la correcta.

Identificador	**Descripción**	**Estimación (puntos de historia)**
HU20	*Como usuario* *Quiero poder acceder a la aplicación* *Con la finalidad de poder utilizar todas sus funcionalidades*	2pts
HU21	*Como usuario* *Quiero poder ver las publicaciones en detalle* *Con el propósito de observar cual es el contenido de una publicación en concreto*	5pts

Cuadro 2.8: Nuevas historias añadidas a la pila del producto.

Observaciones

El desarrollo de este Sprint comenzó con la finalización de las funcionalidades que no había acabado de la pila del producto. Debido a esto, y al igual que en el Sprint 1, se me adelantó que debería mejorar los listados para que estos incluyeran la paginación.

La paginación supuso un problema ya que se necesitaba un paginado que funcionara apropiadamente tanto para una aplicación móvil como para una web, aunque finalmente se llegó a una conclusión adecuada sobre como implementarlo.

Una vez acabé con el apartado de la paginación me dediqué a realizar una refactorización[14] del código del *backend* para hacerlo más eficiente y conseguir que la API REST pueda emplearse más fácilmente.

Reunión de cierre

En la reunión de cierre se realizó una demostración del trabajo desarrollado hasta la fecha, esta demostración se hizo en conjunto entre los dos departamentos de desarrollo y el *Product Owner*.

Esta demostración fue satisfactoria por parte del *backend*, por lo que el Sprint se consideró acabado por mi parte. Esta vez el *frontend* fue capaz de acabar todas las tareas asignadas del Sprint. Esta información queda reflejada en la figura 2.2.

La subida que se observa en la figura es debido a que el *frontend* acabara con las historias programadas para este Sprint y añadiera nuevas historias a completar que originalmente se completarían en Sprints posteriores.

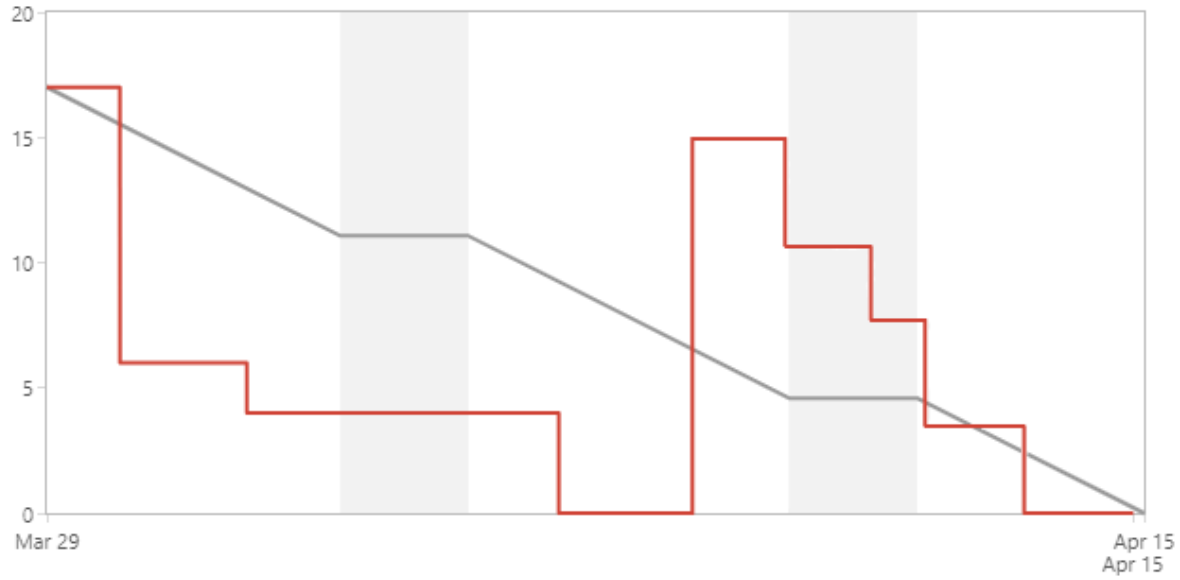

Figura 2.2: Gráfico *Burn Down* del Sprint 2.

2.5.3. Sprint 3

Reunión de planificación

En la reunión de planificación del Sprint 3 se definió el objetivo del mismo, este incluía la vista en detalle tanto de las publicaciones como de las retroalimentación y a su vez la vista en detalle de la vista en detalle de los datos del usuario.

Además de esto, como se me había adelantado en el Sprint anterior, se estipuló que debíamos modificar los listados para incluir la paginación en ellos, ya que estos podrían ser sumamente amplios y empeorar la experiencia del usuario. Además de eso también tendríamos que empezar a manejar adecuadamente los errores, todo esto queda reflejado en el cuadro 2.9.

Identificador	Descripción	Estimación (puntos de historia)
TA03	*Depuración y gestión de errores*	-
TA04	*Realizar paginación en las listas*	-

Cuadro 2.9: Nuevas historias añadidas a la pila del producto.

Observaciones

El desarrollo de este Sprint tuvo un inicio diferente al resto de Sprints. Debido a que ya había acabado con la refactorización del *backend*, mi objetivo era el de iniciar el desarrollo del *frontend* web, pero antes de eso se me comunicó que debería realizar una formación previa sobre el funcionamiento de *ReactJS*.

Una vez esa formación terminó, comencé la implementación del *frontend* web. Mi objetivo era desarrollar funcionalidades de acuerdo con la priorización de las historias preestablecida, y seguí con eso durante el resto del Sprint. El desarrollo de esto fue llevado a cabo sin contratiempos y gran parte del *frontend* web fue completado.

Reunión de cierre

En la reunión de cierre se realizó una demostración del trabajo desarrollado hasta la fecha, esta demostración se hizo en conjunto entre los dos departamentos de desarrollo y el *Product Owner*. Esta demostración fue satisfactoria tanto por parte del *backend* como del *frontend*, web y móvil, por lo que el Sprint se consideró completado. Esta información queda reflejada en la figura 2.3.

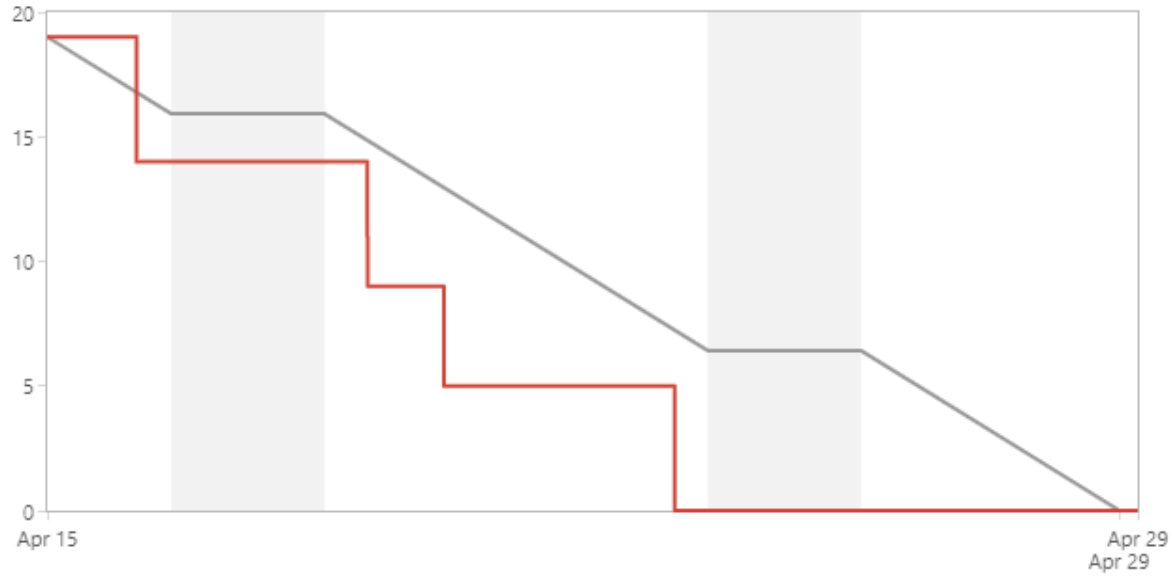

Figura 2.3: Gráfico *Burn Down* del Sprint 3.

2.5.4. Sprint 4

Reunión de planificación

En la reunión de planificación del Sprint 4 se definió el objetivo del mismo, este incluía la notificación de notificar a los usuarios, cuando estos recibieran retroalimentación o valoraciones y añadir nuevos tipos de sugerencias.

Por esto mismo, se demandó que debíamos incorporar un sistema de notificaciones *push* tanto para la aplicación móvil como para la versión web. También se nos pidió que modificáramos la creación de sugerencias para que esta no incluyera únicamente sugerencias sobre categorías sino sobre otras cosas. Esto queda reflejado en el cuadro 2.10. Para implementar las notificaciones de decidió hacer uso de OneSignal[21].

Identificador	**Descripción**	**Estimación (puntos de historia)**
HU22	*Como usuario* *Quiero poder recibir notificaciones cuando alguien interaccione conmigo* *Para poder enterarme y reaccionar a ello*	5pts
HU23	*Como usuario* *Quiero poder hacer sugerencias de todo tipo* *Para poder mejorar la experiencia de usuario o comunicar ciertos bugs a los desarrolladores*	3pts

Cuadro 2.10: Nuevas historias añadidas a la pila del producto.

Observaciones

El desarrollo de este Sprint comenzó con la implementación del resto de historias de la pila del producto que faltaban por implementar en el *frontend* web. Esto fue completado sin mayores complicaciones.

Una vez que esas historias fueron completadas, mi siguiente objetivo fue el de desarrollar el sistema de notificaciones. Esto supuso un ligero problema, tanto por parte del *backend* como del *frontend* web.

Por parte del *bakend* hubo un problema a la hora de encolar las notificaciones, ya que no se podían enviar una vez que se realizara la acción que las provocara, ya que esto supondría una carga excesiva por parte del servidor. Para solucionar esto se diseño una cola de notificaciones

que haría uso de un *cron job*[1], para que cada minuto enviara las notificaciones que no habían sido enviadas.

Por parte del *frontend* web surgió un problema a la hora de suscribirse a las notificaciones, debido a que la documentación sobre eso no era muy exhaustiva, aunque finalmente se logró encontrar como hacerlo y se completó la funcionalidad de las notificaciones.

Tras haber acabado con eso me dediqué a rediseñar la creación de las sugerencias para que incorporaran diferentes tipos de sugerencias. Esto fue completado sin ninguna dificultad.

Finalmente me dediqué a realizar una refactorización del código del *frontend* web para que este fuera más eficiente.

Reunión de cierre

En la reunión de cierre se realizó una demostración del trabajo desarrollado hasta la fecha, esta demostración se hizo en conjunto entre los dos departamentos de desarrollo y el *Product Owner*.

Esta demostración fue satisfactoria por ambos equipos de desarrollo, por lo que el Sprint se consideró completado. Esta información queda reflejada en la figura 2.4.

Además de esto, en este Sprint se completaron todas las funcionalidades restantes de la pila del producto, por lo que el proyecto se consideró completado.

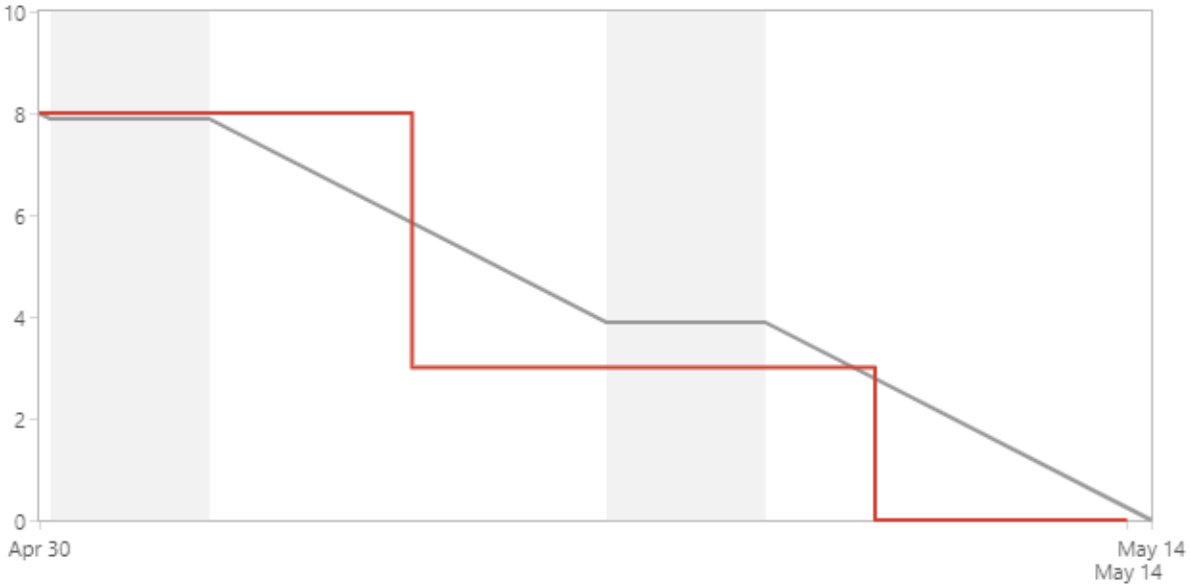

Figura 2.4: Gráfico *Burn Down* del Sprint 4.

[1]Procesos que se ejecutan en segundo plano en intervalos de tiempo

Capítulo 3

Análisis y diseño del sistema

3.1. Análisis del sistema

En esta sección se van a definir los casos de uso del proyecto, para posteriormente especificar cada uno de ellos. Con esto, se busca conseguir realizar una definición exhaustiva del sistema, que permita recoger cuales son sus características principales. Los casos de uso se encargan de describir una secuencia de interacciones entre el sistema y el usuario final, con el objetivo de recoger las diferentes funcionalidades que formaran ese sistema. Estos detallan exhaustivamente dichas interacciones y el comportamiento del sistema frente a ellas.

Los casos de uso aportan una visión que busca centrarse en las especificaciones del sistema por lo que nos ayudan a discernir entre aquello que es verdaderamente importante y aquello que puede ser secundario, es por esto mismo que es una parte muy importante del análisis del sistema.

3.1.1. Casos de uso

A continuación en la figura 3.1 se muestra el diagrama de casos de uso del sistema, donde se recogen todos los casos que intervienen en nuestro proyecto y los actores, entidades o servicios, que interactuaran con ellos.

En este proyecto hay dos actores principales, el Administrador y el Usuario. El administrador se encarga de todas las tareas relativas a la administración del sistema, pero permitiéndole que disfrute de las características principales de la plataforma. El Usuario es aquel que participa de forma activa en la plataforma, es decir, el que aporta contenido. Este actor se divide en dos actores que son el Aprendiz y el Experto.

Tras esto se recogen las especificaciones de los casos mostrados en el la figura 3.1. Esta información queda registrada entre los cuadros 3.1 y 3.3 que se encuentran a continuación, y los cuadros A.1 y A.13 que se encuentran en el anexo A. Tanto el diagrama de la figura 3.1 como las especificaciones fueron realizadas en conjunto entre los dos equipos de desarrollo. Es debido

a eso, que estas especificaciones tienen diferentes autores.

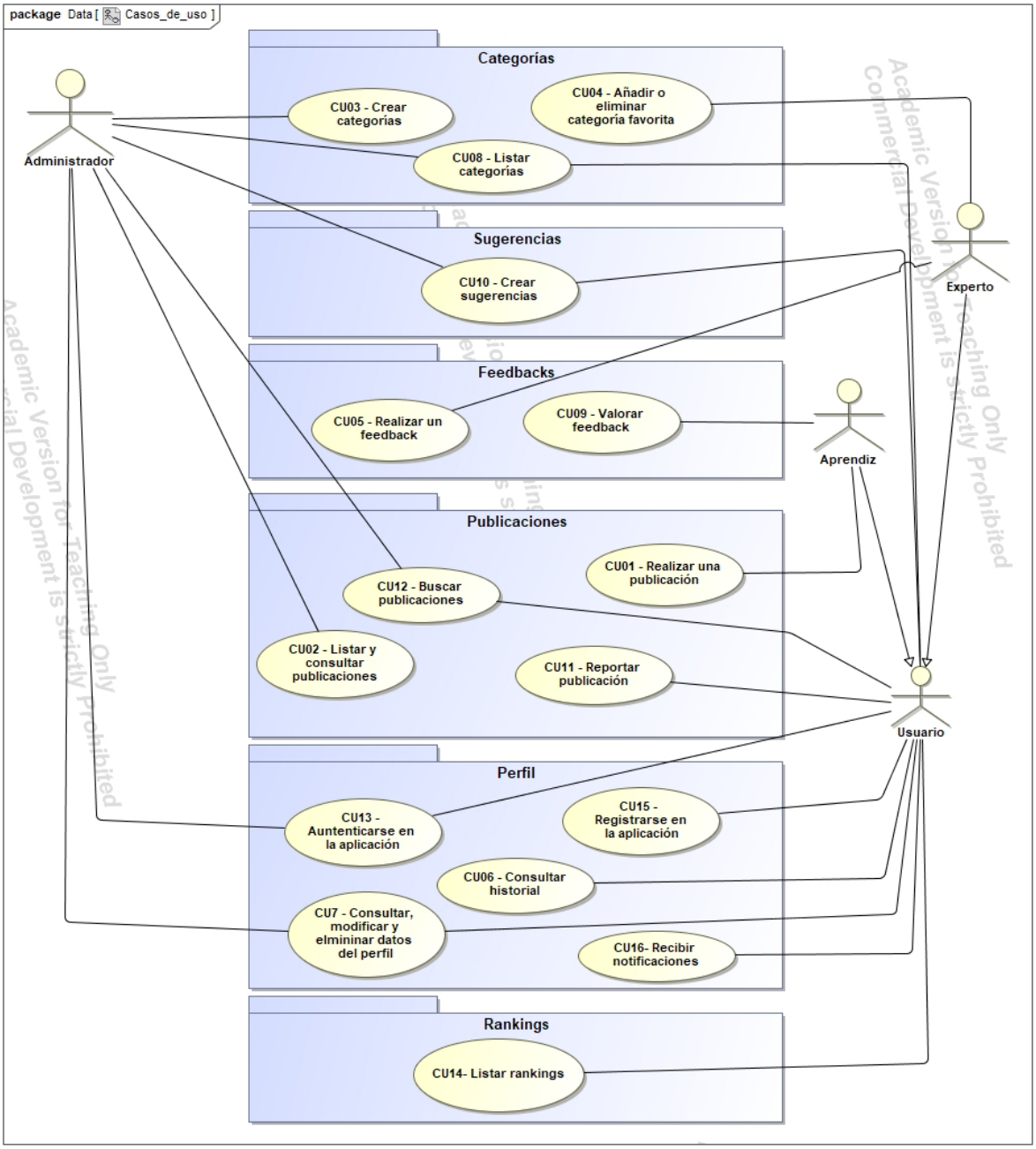

Figura 3.1: Diagrama de casos de uso.

Especificación del caso de uso	
Identificador	CU01
Nombre	Realizar una publicación.
Versión	V01
Autor	Alex Martínez Martínez
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los aprendices puedan crear una publicación para recibir feedback.
Alcance	El sistema ha de permitir que los aprendices creen una publicación a la cual puedan adjuntar archivos de vídeo, imágenes y documentos.
Nivel	Tarea principal
Actor principal	Aprendiz
Actores secundarios	N/A
Relaciones	CU02, CU11, CU12
Precondición	El aprendiz ha de estar registrado en la aplicación.
Condición fin con éxito	La publicación ha sido creada y añadida a la lista de publicaciones.
Condición fin con fracaso	La publicación no se crea y no se añade a la lista de publicaciones.
Trigger	El aprendiz desea recibir retroalimentación sobre un tema concreto.
Secuencia normal	**Acción**
1	El aprendiz se autentifica en la aplicación.
2	El aprendiz accede a la vista de crear publicación.
3	El aprendiz rellena toda la información relativa a la publicación.
4	El aprendiz adjunta archivos a la publicación.
5	El aprendiz realiza la publicación.
6	El sistema almacena la publicación y notifica al aprendiz de que esta ha sido creada satisfactoriamente.
Excepción en 5 - Categoría no encontrada	**Excepción**
1	El sistema no almacena la publicación y notifica el error.
2	El aprendiz rellena toda la información relativa a la publicación.
3	El aprendiz adjunta archivos a la publicación.
4	El aprendiz realiza la publicación.
5	El sistema almacena la publicación y notifica al aprendiz de que esta ha sido creada satisfactoriamente.
Excepción en 5 - Cancelar operación	**Excepción**
1	El aprendiz cancela la operación.
Frecuencia esperada	100 veces al día
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro 3.1: Especificación del caso de uso CU01.

Especificación del caso de uso	
Identificador	CU02
Nombre	Listar y consultar publicaciones.
Versión	V01
Autor	Carlos Mora Hernández
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los usuarios puedan listar y consultar las publicaciones creadas.
Alcance	El sistema ha de permitir que los usuarios puedan listar y consultar las publicaciones creadas, consultado tanto su información como los archivos adjuntados.
Nivel	Tarea principal
Actor principal	Experto, Aprendiz
Actores secundarios	Administrador
Relaciones	CU01, CU11, CU12
Precondición	El usuario ha de estar registrado en la aplicación.
Condición fin con éxito	Las publicaciones se listan y se puede consultar individualmente.
Condición fin con fracaso	Las publicaciones no se listan y no se puede consultar individualmente.
Trigger	El aprendiz o experto desea consultar una publicación previa para ver de que trata.
Secuencia normal	**Acción**
1	El usuario se autentifica en la aplicación.
2	El usuario accede a la vista listar publicaciones.
3	El sistema lista todas las publicaciones.
3.1	El usuario accede a una publicación concreta.
3.2	El sistema devuelve la información relativa a dicha publicación.
Excepción en 3 - Error de servidor	**Excepción**
1	El sistema no lista las publicaciones y muestra un mensaje de lista vacía.
2	El usuario accede a la vista listar publicaciones.
3	El sistema lista todas las publicaciones.
3.1	El usuario accede a una publicación concreta.
3.2	El sistema devuelve la información relativa a dicha publicación.
Excepción en 3.2 - Publicación no encontrada	**Excepción**
1	El sistema no devuelve la información sobre la publicación y notifica el error.
2	El sistema lista todas las publicaciones.
2.1	El usuario accede a una publicación concreta.
2.2	El sistema devuelve la información relativa a dicha publicación.
Frecuencia esperada	200 veces al día
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro 3.2: Especificación del caso de uso CU02.

Especificación del caso de uso	
Identificador	CU05
Nombre	Realizar un feedback.
Versión	V01
Autor	Alex Martínez Martínez
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los expertos puedan realizar feedbacks sobre publicaciones creadas previamente.
Alcance	El sistema ha de permitir que los expertos puedan realizar feedbacks sobre publicaciones creadas previamente aunque no sean expertos en la categoría de la publicación permitiéndole adjuntar archivos de imagen, vídeo y documentos.
Nivel	Tarea principal
Actor principal	Experto
Actores secundarios	N/A
Relaciones	CU09
Precondición	El experto ha de estar registrado en el sistema.
Condición fin con éxito	El feedback ha sido creado y añadido a la lista de feedbacks.
Condición fin con fracaso	El feedback no se crea y no se añade a la lista de feedbacks.
Trigger	Se quiere dar retroalimentación a la publicación de algún aprendiz.
Secuencia normal	**Acción**
1	El experto se autentifica en la aplicación.
2	El experto accede a la vista de aportar feedback.
3	El experto rellena toda la información relativa al feedback.
4	El experto adjunta archivos al feedback.
5	El experto crea el feedback.
6	El sistema almacena el feedback y notifica al experto que este ha sido creado satisfactoriamente.
Excepción en 5 - Cancelar operación	**Excepción**
1	El experto cancela la operación.
Excepción en 6 - Publicación no encontrada	**Excepción**
1	El sistema no almacena el feedback y notifica el error.
2	El experto rellena toda la información relativa al feedback.
3	El experto adjunta archivos al feedback.
4	El experto crea el feedback.
5	El sistema almacena el feedback y notifica al experto que este ha sido creado satisfactoriamente.
Frecuencia esperada	100 veces al día
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro 3.3: Especificación del caso de uso CU05.

3.1.2. Diagrama de clases

A continuación se define el diagrama de clases, que representa la estructura del sistema, es decir, todos los componentes que forman y como estos se relacionan entre ellos. Todo esto queda recogido en la figura 3.2

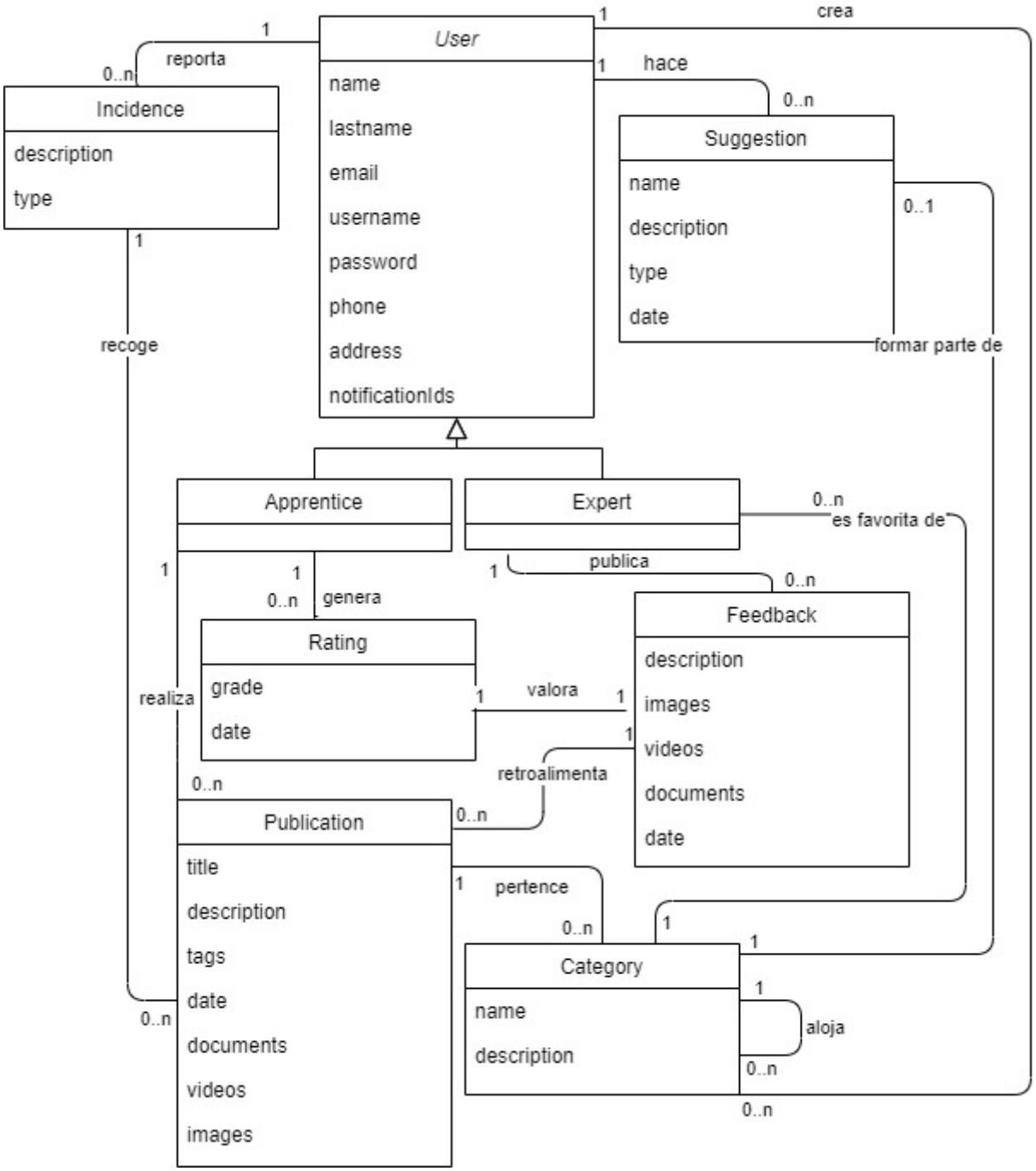

Figura 3.2: Diagrama de clases.

Cabe mencionar que los administradores también son usuarios del sistema, es por eso mismo que no he diseñado ninguna entidad adicional para estos usuarios, y es por eso mismo mismo

que *User* se relaciona con *Category*.

Además de esto las notificaciones forman parte de una entidad que surge de las asociaciones valora y retroalimenta. Estas notificaciones son elementos complementarios del sistema, es por eso mismo que, no los he adjuntado en este diagrama.

3.2. Diseño de la arquitectura del sistema

En esta sección se van a describir los diferentes componentes del sistema y como estos se relacionan entre ellos. Para recoger esto se va a hacer uso de un diagrama de arquitectura, donde este recogerá todos los componentes que forman tanto el *backend* como el *frontend* web. Todo esto queda recogido en la figura 3.3. Cabe mencionar que el diseño del conjunto del proyecto sigue el modelo clásico cliente/servidor.

En este diagrama, se puede observar como se han estructurado el *frontend* y el *backend* a nivel de componentes. En cuanto al *backend* todo su desarrollo se ha definido entorno a la API REST, mientras que en el *frontend* se han definido una serie de módulos, que representan las funcionalidades de las que dispondrá el sistema, y algunas de ellas estarán envueltas por unas *Store* que aportarán información general al sistema. La explicación en detalle de todos estos componentes vendrá dada en el capítulo siguiente.

Las *Store* hacen uso de una herramienta de *ReactJS* llamada *Context*, que también será expuesto en el capítulo siguiente, cuya tarea es la de aportar un estado común a todos los módulos que envuelve, para que estos puedan comunicarse entre sí incluso si son totalmente independientes uno del otro.

Cada módulo que compone el *frontend* web va estar formado por 3 componentes diferentes, como se muestra en la figura 3.4, que son:

- *Screen* (Pantalla): Se encarga de renderizar lo que se va a mostrar en la pantalla.
- DataContainer (Contenedor de Datos): Se encarga de almacenar todos los datos necesarios.
- View (Vista): Se encarga de mostrar los componentes con los que el usuario puede interactuar.

Todo esto se explicará más en detalle en el capítulo siguiente, ahora solo he expuesto de una manera muy acotada las tareas que han de realizar estos componentes.

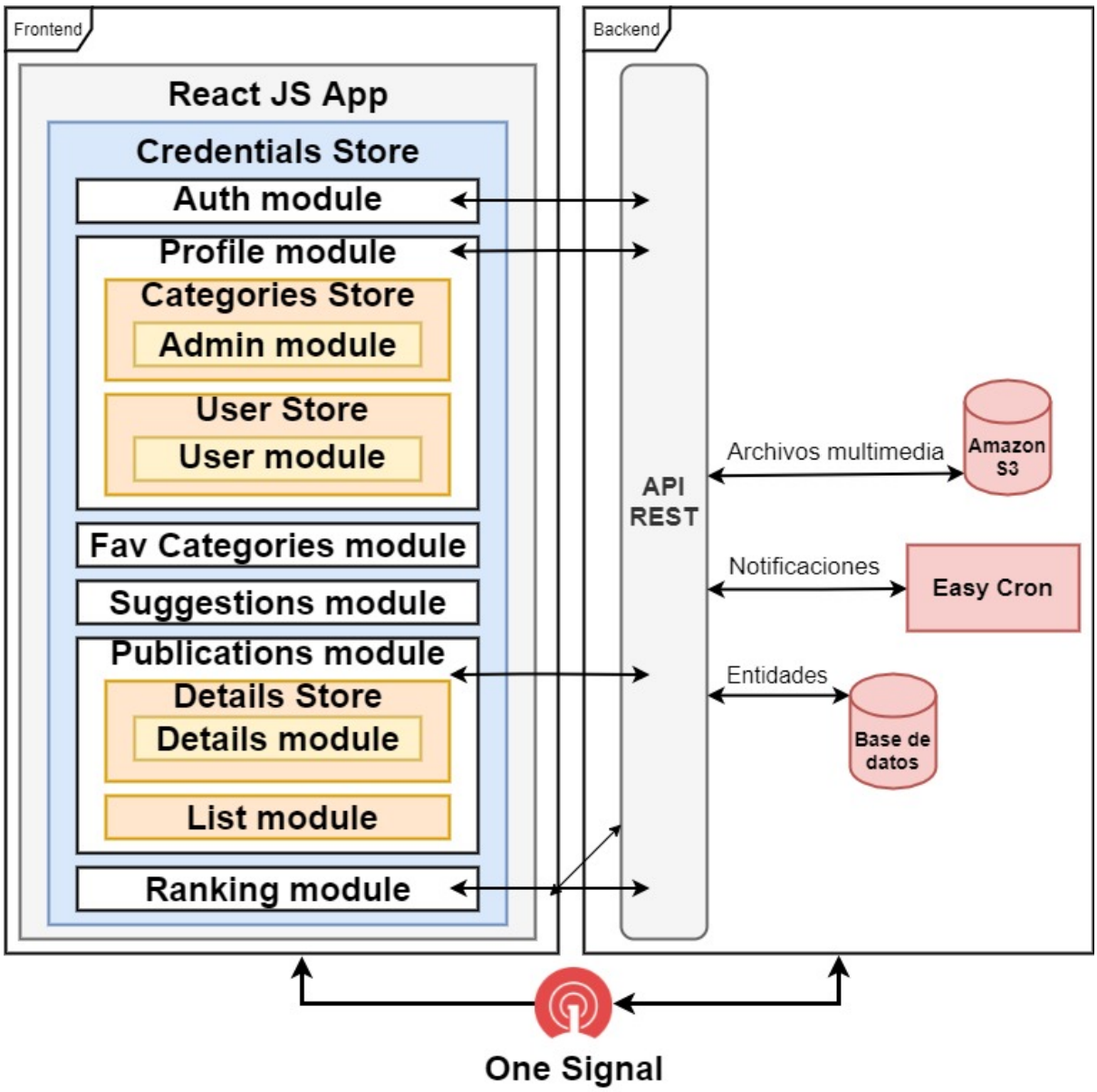

Figura 3.3: Diagrama de arquitectura del sistema.

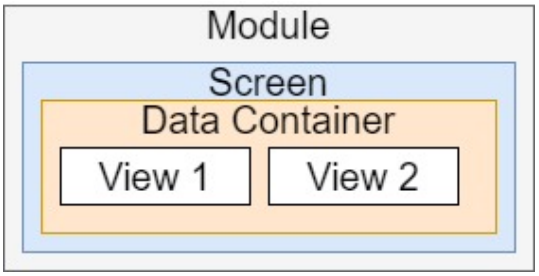

Figura 3.4: Módulo del sistema en detalle.

3.2.1. Diagrama Entidad-Relación

A continuación se va a mostrar el diagrama entidad-relación que recoge toda la información relativa a la implementación de la base de datos. En este encontraremos todas las entidades que fueron definidas en la figura 3.2, pero estas ahora serán más extensas, ya que se incluirán las relaciones de la mismas a nivel de la base de datos.

Es necesario puntualizar que existen ciertas diferencias entre el diagrama de clases y el de entidad-relación, esto es debido que a nivel de base de datos ciertas relaciones no eran necesarias, por lo que se han suprimido.

También han aparecido dos nuevas entidades, la de *Notification*, que almacena los datos relativos a las notificaciones; y la de *NoActiveUser* que almacena los datos de los usuarios que han decidido darse de baja en la aplicación. Todo esto queda reflejado en la figura 3.5.

En el capítulo siguiente se expondrá más en detalle como se ha creado la base de datos y como afecta esto al desarrollo del sistema.

3.3. Diseño de la interfaz

Como este proyecto se va a realizar de manera conjunta entre los departamentos web y móvil, el diseño de la interfaz debía ser común, o al menos similar, en ambos *frontend*, web y móvil. Debido a esto, y a que el *frontend* móvil estaba más avanzado que el web, yo tenia que adaptarme al diseño de las interfaces de la aplicación móvil.

Para el diseño de las interfaces se tuvieron en cuenta las 3 reglas de oro del diseño de interfaces de Theo Mandel[12], que dictamina lo siguiente:

- Dar el control al usuario.
- Reducir la carga de memoria del usuario.
- Mantener la consistencia de la interfaz.

La primera regla define que el usuario ha de ser quien posea el control tanto para interactuar con todos los componentes de la interfaz como para hacer y deshacer a su antojo. La segunda regla aboga por que el usuario no necesite recordar información a largo plazo sobre sus interacciones con la interfaz. La tercera y última busca que la interfaz en todos sus puntos mantenga un orden inicial y que ni su disposición ni su diseño varíen a lo largo del tiempo.

3.3.1. Guía de estilos

Todas las interfaces del *frontend* web han sido diseñadas de una misma manera en la que se ha hecho uso de *Bootstrap*. Se ha intentado en todo momento que el diseño sea lo más simple posible y que el flujo de acciones sea lo más coherente posible.

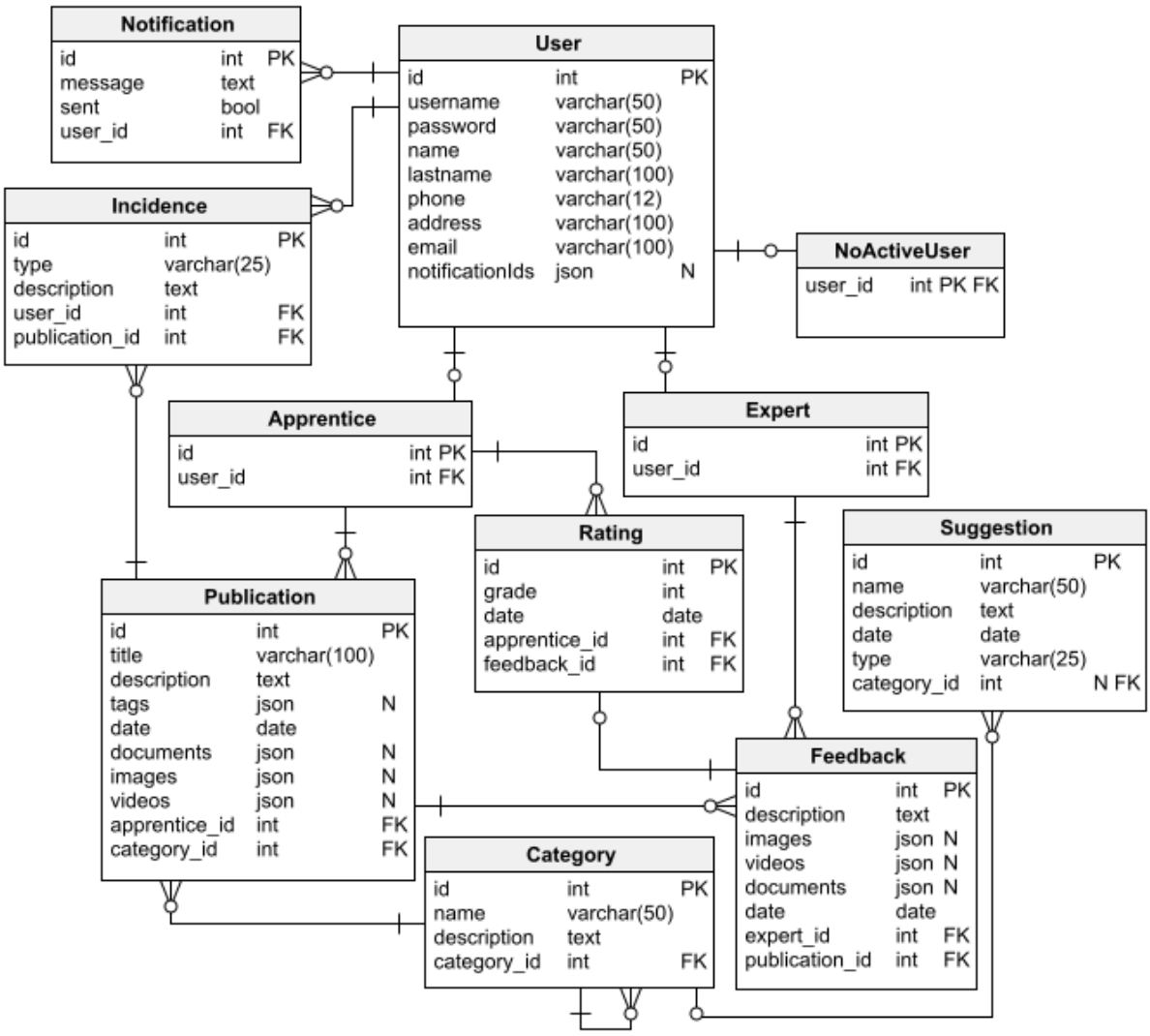

Figura 3.5: Diagrama Entidad-Relación.

Paleta principal

La paleta de colores empleada define los colores basándose en el tipo de acción que van a representar. De la paleta estándar que ofrece *Bootstrap*, recogidas en la figura 3.6 no se han empleado las tonalidades, *Dark* y *Warning*, ya que no se consideró que fueran necesarias para aportar información sobre las acciones de la plataforma y también ya se disponían de suficientes variantes para diferenciar todos los tipos de información, más variantes podrían haber supuesto una carga de colores excesiva para el usuario.

Botones

En cuanto a los botones se han empleado diferentes estilos para ellos en función de la acción que debían desempeñar. Como puede observarse en la figura 3.7 se han empleado 3 colores

Figura 3.6: Paleta de colores.

diferentes, estos colores definen que acción van a desempeñar. Para las acciones de confirmación se ha usado la variante *Primary*, para las acciones de cancelación se ha empleado la variante *Secondary* y para las acciones de eliminación se ha utilizado la variante *Danger*.

También hay que destacar que los botones pueden presentar dos formas diferentes, la primera es la forma que se muestra en la figura 3.7, una forma rectangular con los bordes redondeados, esta forma se ha usado en aquellas partes en la que usar la otra forma podría perturbar el diseño de la vista. La segunda forma que presenta es la circular, esta forma se ha empleado para adaptar el estilo del *frontend* móvil, en aquellas vistas en las que pueda encajar satisfactoriamente.

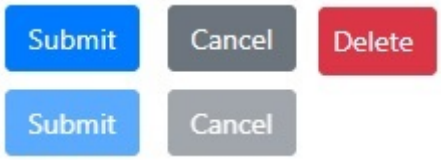

Figura 3.7: Diseño de los botones.

Alertas

En cuanto a las alertas, al igual que los botones, se han empleado diferentes estilos, donde cada estilo indica una característica del mensaje de información que aporta dicha alerta. Como puede observarse en la figura 3.8 se han requerido 3 colores diferentes. Para los mensajes de éxito, es decir, aquellos que representan que la acción se ha completado sin errores se ha usado la variante *Success*, para los mensajes de error, es decir, aquellos que representan que la acción no se ha completado sin errores se ha utilizado la variante *Danger*, finalmente para los mensajes que representan que una acción esta en proceso de ejecución se ha empleado la variante *Info*, únicamente ha sido empleada en acciones que pueden llegar a tardar en exceso,.

Figura 3.8: Diseño de las alertas.

Iconografía

Con respecto al tema de los iconos que han sido empleados en el *frontend* web, estos no pertenecen a los estilos de *Bootstrap*. Para el diseño de la interfaz de este proyecto se han empleado los iconos de la librería Semantic UI [11] junto a su librería de estilos css. Algunos de los iconos empleados en la interfaz quedan reflejados en la figura 3.9

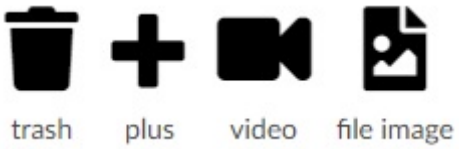

Figura 3.9: Diseño de las iconos.

Capítulo 4

Implementación y pruebas

4.1. Detalles de implementación

En esta sección se va a exponer en detalle como se ha desarrollo el proyecto. Por una parte, expondré la implementación del *backend* y por otra parte, expondré la implementación del *frontend* web. Con esto explicaré las decisiones de implementación tomadas, así como los patrones o estrategias que he seguido para desarrollarlos.

Una vez completado eso, mostraré los resultados de la implementación, en caso del *frontend* web serán las vistas finales con las que el usuario podrá interactuar, en caso del *backend* será la interfaz de *Swagger*[25] que recoge todos los *endpoints*[24] de la API y te permite probarlos.

Finalmente expondré todos los problemas que me han surgido durante el desarrollo del proyecto, así como las soluciones a los mismos.

4.1.1. Patrones de diseño

Antes de exponer como se ha realizado el desarrollo del proyecto, voy a exponer los patrones de diseño que he empleado y su definición.

Inyección de dependencias

La inyección de dependencias[4] es uno de los patrones de diseño más empleado que se centra en evitar que los diferentes componentes que forman una aplicación dependan entre sí de una manera muy directa.

Para lograr que este patrón funcione correctamente se necesita hacer uso de :

- Emplear abstracciones: Uno de los requerimientos para el desarrollo de este patrón es

eliminar la dependencia entre componentes.Para lograr esto se deben crear abstracciones, es decir, crear interfaces que permitan esconder esa dependencia.

- Proporcionar el componente de la dependencia a través de un contenedor: Crear un contenedor que cree los objetos que originan la dependencia y otorgarlos al objeto que lo necesita.

A continuación, en la figura 4.1, queda reflejado como funciona este principio.

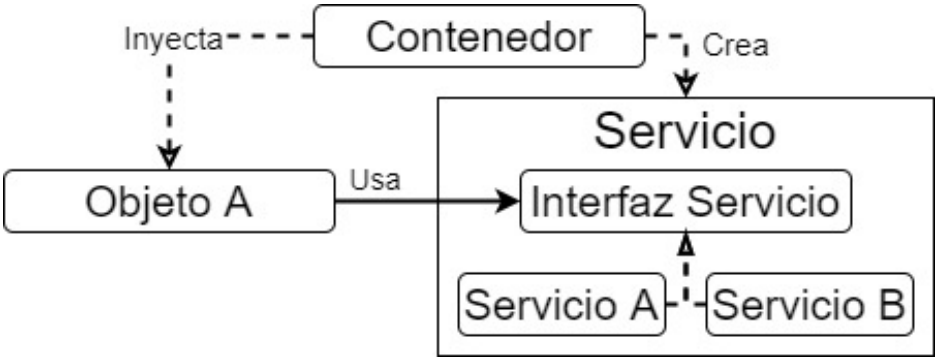

Figura 4.1: Patrón de diseño Inyección de Dependencias.

4.1.2. Estructura y desarrollo

En este apartado voy a exponer como se han estructurado y como se han desarrollado tanto el *backend* como el *frontend* web. Se van a detallar los detalles de como se ha definido cada parte del proyecto.

Backend

La estructura general del *backend* queda refleja en la figura 4.2. El backend ha sido realizado con PHP y *Symfony*, por lo que la estructura que se ve en la figura ha sido completamente generada automáticamente, excepto la carpeta *Utils*, que es donde se recogen clases de ayuda que permiten facilitar ciertas acciones.

A continuación voy a exponer el significado de cada una de las carpetas que componen la estructura del *backend*:

- *bin*: En esta carpeta se definen archivos de configuración de PHP, es algo en lo que no he intervenido directamente.
- *config*: En esta carpeta quedan recogidos todos los archivos de configuración del proyecto, es donde se definen como todos los diferentes paquetes externos se van a comportar.
- *migrations*: En esta carpeta es donde se crean todos los archivos de configuración de la base de datos. Para la base de datos se ha empleado *Doctrine Manager*[7] por lo que los archivos de migración de la base de datos se generan automáticamente.

- *public*: Esta carpeta es la que contempla que la aplicación pueda visualizarse, es la encargada de cargar la vista del proyecto. En nuestro caso solo se encargará de renderizar la interfaz de *Swagger*.
- *src*: En esta carpeta es donde se va a definir el código fuente del *backend*, es decir, es donde se desarrollarán todas las funcionalidades. También encontraremos la definición de las diferentes entidades que conformarán la base de datos y los repositorios de las mismas. Finalmente, encontraremos los servicios y componentes de ayuda.

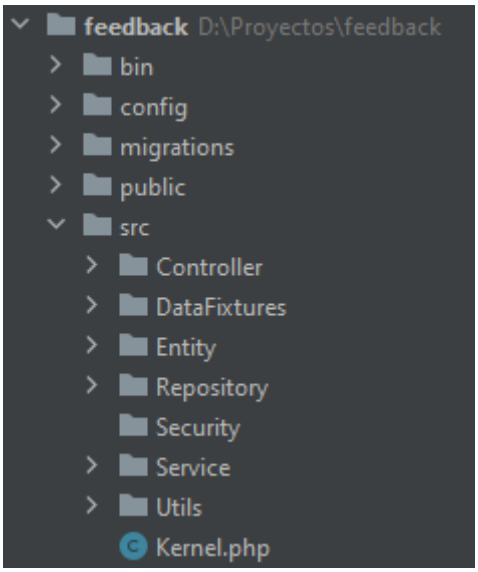

Figura 4.2: Estructura del *Backend*.

A continuación voy a exponer como esta conformada la carpeta *src*, además voy a indicar que incluyen las carpetas que la conforman.

La carpeta *Controller* (Controlador) es donde se definen todos los diferentes *endpoints* que conforman la API REST del *backend*. Cada controlador únicamente debe interaccionar con la entidad que este engloba. Habrá un controlador por cada entidad que recoge la base de datos, y también habrá un controlador por cada funcionalidad extra que no se recoja como tal en la base de datos. Pongamos el ejemplo de listar las publicaciones que se puede observar en el fragmento de código 4.1.

```
1  <?php
2
3  namespace App\Controller;
4
5  //imports
6
7  class PublicationController extends AbstractController
8  {
9      private Serializer $serializer;
10
11     public function __construct(SerializerService $serializerService)
12     {
13         $this->serializer = $serializerService->getSerializer();
```

```
14      }
15
16      #[Route('/api/publication', name: 'publication_get', methods: ['GET'])]
17      /**
18       * Definicion del endpoint para swagger
19       */
20      public function getPublications(Request $request): Response
21      {
22          //Obtienes los datos de la peticion
23
24          //Obtienes el listado de las publicaciones
25
26          //Serializas la respuesta
27          $data = $this->serializer->serialize($publications, 'json', [
28              AbstractNormalizer::GROUPS => ['publications']
29          ]);
30
31          //Creas la respuesta y la devuelves
32          return new JsonResponse($response, 200);
33      }
34  }
```

Código Fuente 4.1: Clase *Publication Controller*

Como puede observarse, el controlador recibe como parámetro un servicio que yo he definido. Este servicio se encarga de devolver el serializador[23] ya configurado, la función del serializador es la de descodificar la petición que recibe el controlador y el de codificar la respuesta para poder enviarla. Este servicio viene dado por el propio *Symfony*, que hace uso de la inyección de dependencias.

Una vez el controlador esta definido, hay que implementar el método al que el *endpoint* correspondiente se dirigirá. Lo primero que hay que definir es la ruta del *endpoint*, indicándole un nombre único que la identificará y a que métodos HTTP[18] (en inglés, *Hypertext Transfer Protocol*) esta sujeto. Tras esto tenemos que definir como será el *endpoint* a nivel de parámetros que recibe, todas las posibles respuestas que puede dar y tipo de seguridad.

Tras esto ya podemos realizar todas las operaciones que queramos, primero hay que obtener los datos de la petición, en caso de ser necesario se tendrían que deserializar, tras esto realizamos toda la gestión de los datos y finalmente creamos una respuesta para mandarla de vuelta.

La carpeta *DataFixtures* (Accesorio de datos) es donde se van a definir los datos base de las entidades de la base de datos, es decir, es donde crearemos objetos base que estarán almacenados en la base de datos desde un principio y que se fuerzan dentro de ella.

La carpeta *Entity* (Entidad) es donde se describen las entidades que forman parte de la base de datos, cada entidad tiene su propia definición separada del resto. De estas entidades se definen todos sus atributos, todas las relaciones que presentan y además se definen *setters* y *getters* para cada atributo de la misma. Pongamos el ejemplo de la entidad *Publication*, que puede observarse en el fragmento de código 4.2.

```
1   <?php
```

```php

namespace App\Entity;

//imports

/**
 * @ORM\Entity(repositoryClass=PublicationRepository::class)
 */
class Publication
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     * @OA\Property(type="integer")
     * @Groups({"publications"})
     */
    private $id;

    /**
     * @ORM\ManyToOne(targetEntity=Category::class)
     * @ORM\JoinColumn(nullable=false)
     * @OA\Property(ref=@Model(type=Category::class))
     */
    private $category;

    //resto de atributos

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getCategory(): ?Category
    {
        return $this->category;
    }

    public function setCategory(?Category $category): self
    {
        $this->category = $category;

        return $this;
    }
    //resto de getters y setters
}
```

Código Fuente 4.2: Entidad *Publication*

Lo primero que podemos observar es que hay 3 tipos diferentes de anotaciones. La primera es la que comienza con @ORM, esta anotación está asociada a la base de datos, son las anotaciones que usa Doctrine para saber como debe de crear y almacenar esos atributos. La segunda es la que empieza con @OA, esta anotación está asociada a la documentación de la API REST y es la que establece como son los campos de la entidad. La tercera y última es la que comienza

con @Groups, esta anotación se encuentra asociada al serializador y a la documentación, si este grupo esta presente en la documentación de la API o en la configuración del serializador, solo se emplearan los atributos que contengan esta anotación.

La carpeta *Repository* (Repositorio) es donde se definen llamadas personalizadas a la base de datos. Todos los repositorios definen por defecto 4 consultas a la base de datos, pero se pueden definir consultas extra para poder responder adecuadamente a las necesidades que se necesiten cumplir. Expongamos el ejemplo del *Publication Repository*, que se puede observar en el fragmento de código 4.3.

```
1  <?php
2
3  namespace App\Repository;
4
5  //imports
6
7  class PublicationRepository extends ServiceEntityRepository
8  {
9      //constructor por defecto
10
11     public function findAllGreaterId($cursor, $filter): array
12     {
13         $entityManager = $this->getEntityManager();
14
15         $query = $entityManager->createQuery(
16             'SELECT p
17                     FROM App\Entity\Publication p
18                     JOIN p.category c
19                     JOIN p.apprentice a
20                     WHERE p.id < :cursor AND
21                     (LOWER(c.name) LIKE :filter
22                     OR LOWER(p.title) LIKE :filter
23                     OR LOWER(a.username) LIKE :filter
24                     OR LOWER(p.tags) LIKE :filter)
25                     ORDER BY p.id DESC'
26         )
27             ->setParameter('filter', '%'.$filter.'%')
28             ->setParameter('cursor', $cursor);
29
30         return $query->getResult();
31     }
32 }
```

Código Fuente 4.3: Clase *Publication Repository*

Como puede observarse el definir una consulta propia requiere que esta sea definida en el lenguaje DQL que es el que emplea *Doctrine*. En esencia es muy similar al lenguaje SQL, primero tenemos que decirle al *Entity Manager* (Administrador de las entidades) que nos cree una sentencia para la base de datos. Esta sentencia puede formarse como una quiera y si requiere de parámetros, estos deben de proporcionarse como se observa en las lineas 27 y 28. Hay un gran número de opciones extra a la hora de crear la sentencia, como pueden ser *JOINS*, se le puede indicar el número máximo de resultados, por que resultado debe empezar, entre otras

muchas.

La carpeta *Security* (Seguridad) es donde se definen los criterios de seguridad de acceso a la aplicación, es decir, es donde defines que datos necesita un usuario para autenticarse. En mi caso he usado los valores por defecto de nombre de usuario y contraseña, por lo que no he definido ningún criterio personalizado.

Las carpetas *Service* (Servicios) y *Utils* (Útiles) son aquellas que definen clases de ayuda para desarrollar todas las funcionalidades que deseemos. Estas clases simplemente son complementos que permiten extraer funcionalidades de las clases controladores para poder limpiar el código y que este sea más legible y pueda atender al patrón de la inyección de dependencias.

Frontend

La estructura general del *frontend* queda reflejada en la figura 4.3. El *frontend* ha sido desarrollado con *ReactJS*, es por esto que la estructura principal ha sido generada por el propio *ReactJS*, pero todo aquello que se encuentra en la carpeta *src* ha sido definido y estructurado por mi.

A continuación voy a exponer el significado de cada una de las carpetas que componen la estructura del *frontend*:

- *public*: Esta carpeta es la que contempla que la aplicación pueda visualizarse, es la encargada de cargar la vista del proyecto. En nuestro caso se encargará de renderizar la vista o vistas en las que nos encontremos durante la navegación.

- *src*: En esta carpeta es donde se va a definir el código fuente del *frontend*, es decir, es donde se desarrollarán todas las funcionalidades que estarán presente en los diferentes módulos. Además de esto definiremos una serie de componentes de apoyo, contextos y entidades que facilitaran el manejo de información. También se definirá una API base para realizar peticiones y un rutado básico de la aplicación.

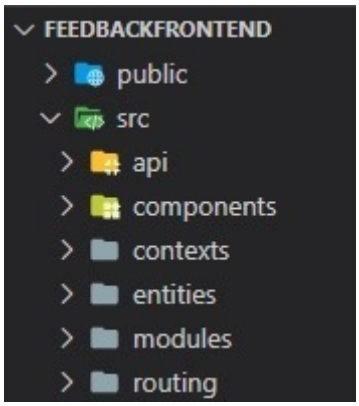

Figura 4.3: Estructura del *Frontend*.

La carpeta api es donde se define la ruta a la API REST por defecto. Las peticiones se van a realizar mediante axios[5] por lo que tendremos que definir una URL (Localizador de Recursos Uniforme) base a la que se realizarán las peticiones. En mi caso también he definido un interceptor, este se encarga, como su nombre indica, de interceptar las peticiones y aplicarles una configuración adicional. Esto se ha aplicado para poder proporcionarles el token de identificación a las peticiones. En el fragmento de código 4.4 se puede observar la definición de la api.

```
1  //imports
2
3  const api = axios.create({
4      baseURL: 'base_url',
5      headers: {
6          /*configuracion base*/
7      }
8  })
9
10 api.interceptors.request.use((config: AxiosRequestConfig) => {
11     config.headers = {...config.headers, /*nueva configuracion*/}
12     return config
13 }, (err: any) => {
14     return Promise.reject(err)
15 })
16
17 export default api
```

Código Fuente 4.4: Clase api

La carpeta *components* (Componentes) es donde se definen todos los componentes que son reutilizables en diferentes lugares de la aplicación. En mi caso los componentes reutilizables que he definido han sido los diferentes tipos de componentes de un formulario, el paginado, la barra de búsqueda, el *header* (encabezado) de la aplicación y por último la *layout* (disposición) de la aplicación.

La *Layout* la empleo para discernir como debo renderizar los componentes en la pantalla, por ejemplo, supongamos que estoy en la pantalla de registro, si no queremos que se muestre el *Header* lo indicamos en la *Layout* y este no se mostrará.

La carpeta *contexts* (Contextos) es donde se definen los contextos que se emplearán en la aplicación. Los contextos en *ReactJS* proporcionan una forma de compartir información entre diferentes componentes sin que estos tengan que estar completamente relacionados. Habitualmente la forma de compartir información entre componentes es usar valores compartidos entre padres e hijos pero esto puede ser muy limitante. En el fragmento de código 4.5 se puede observar la implementación de un contexto.

```
1  //imports
2
3  export interface CredentialsData {
4      //datos que presentara el contexto
5  }
6
7  const CredentialsContext = React.createContext<CredentialsData>({
```

```
8       //datos del contexto
9   })
10
11  export const CredentialsProvider = CredentialsContext.Provider
12
13  export interface CredentialsStoreProps {
14      children: JSX.Element
15  }
16
17  export const CredentialsStore = (props: CredentialsStoreProps) => {
18
19      //definicion de los metodos que tratan los datos del contexto
20
21      return (
22          <CredentialsContext.Provider value={{
23              //datos del contexto
24          }}>
25              {props.children}
26          </CredentialsContext.Provider>
27      )
28  }
29
30  export default CredentialsContext
```

Código Fuente 4.5: Contexto *Crenditals*

La forma habitual de diseñar un contexto es implementar exclusivamente la definición tanto del propio contexto como la de su proveedor, pero esto puede suponer problemas si hay varios componentes que consumen y modifican el contexto ya que la información podría corromperse. Es por esto mismo que se define una *store* (tienda) del contexto, que definirá una serie de métodos que sus hijos, es decir, todos los componentes que consuman el contexto emplearan, para que el cambio de la información se haga únicamente en la tienda del contexto y la información no se corrompa.

La carpeta *entities* (Entidades) es donde se definen todas las entidades que se emplean en la aplicación, estas corresponden a todos los tipos de devolución de las peticiones que realizamos a la API REST del *backend*. Estas entidades se emplean para el tipado de axios en las peticiones que este realiza. Además de definir las entidades también se definen los tipos de valores que se requerirán en los formularios en los que esas entidades intervengan.

La carpeta *modules* (Módulos) es donde se definen todos los módulos que se emplearan en la aplicación, estos módulos representan las funcionalidades sobre una entidad de la aplicación, como ejemplo de esto podemos diferenciar los módulos de perfiles, de publicaciones, de categorías, entre otros.

Dentro de cada módulo encontramos una estructura especial que lo define. Un módulo esta compuesto por una *Screen* (Pantalla) principal, cuyo objetivo es el de contener la información del módulo que se esta reflejado en la interfaz. Dentro de cada *Screen* encontramos los *DataContainers* (Contenedores de datos), estos se encargan de definir todos los datos que se van a requerir en la vista que tiene asociada, dentro de esto datos también encontramos el consumidor del contexto. Finalmente cada *DataContainer* tiene asociada una *View* (Vista) propia, esta vis-

ta se encarga de renderizar los componentes que la conforman, sirve como intermediario entre el *DataContainer* y los diferentes componentes que se definen en el módulo. En el fragmento de código 4.6 se puede observar la definición de estos elementos, por simplificación se han juntado en un único fragmento, pero para su correcta implementación deberían definirse por separado.

```
//SCREEN
const Screen = () => {
    return <> <DataContainer /> </>
}
export default Screen

//DATA CONTAINER
const DataContainer = () => {
    const [ data, setData ] = useState</*tipo*/>(/* valor por defecto */)
    const context = useContext(/* contexto */)
    const {/* opiones formulario */} = useForm</* tipo formulario */>()
    const ref = useRef</* tipo */>(/* valor por defecto */)

    //metodos

    useEffect(() => {
        //efecto
    }, [/* momentos de rerenderizado */])

    return <View data={data}/>
}
export default DataContainer

//VIEW
interface ViewProps {
   //datos de la vista
   data: string
}
const PublicationListView: FunctionComponent<ViewProps> = ({data}) => {
    return <div> <Component data={data}/> </div>
}
export default View

//COMPONENT
interface ComponentProps {
    //datos del componente
    data: string
}
const Component: FunctionComponent<ComponentProps> = ({data}) => {
    return <> {data} </>
}
export default Component
```

Código Fuente 4.6: Base para definir un módulo

Como puede observarse, el flujo de información va desde la *Screen* hasta el *Component*. Cabe destacar que en este proyecto he hecho uso de los *hooks* de *ReactJS*, estos te permiten usar las diferentes características que se encuentran en *ReactJS* de una manera más sencilla y cómoda,

con características me refiero a estado, renderizado, referencias, entre otras.

Los *hooks* que he empleado a lo largo de aplicación son 5, 4 de ellos son nativos de *ReactJS* y el 5 es un *hook* creado por la comunidad.

- *useState*: Este se encarga de definir un estado, con estado me refiero a un valor que se quiere conservar en el *DataContainer*. Este *hook* permite tanto acceder al valor del estado como modificarlo.

- *useContext*: Este se encarga de permitir que el *DataContainer* consuma los valores del contexto y todos los métodos que este define.

- *useRef*: Este se encarga de otorgar una referencia a un componente de la vista, para poder acceder a él desde cualquier parte y consultar toda su información.

- *useEffect*: Este se encarga de la renderización, permite definir que hacer cada vez que se debe renderizar un *DataContainer*, y que valores accionan la renderización del mismo.

- *useForm*: Este se encarga de definir un formulario, que permite hacer uso de una serie de funciones para diferentes acciones. Por ejemplo, este *hook* nos permite registrar una serie de atributos al formulario, así como observar sus cambios, entre otras cosas.

La carpeta *routing* (Rutado) define las rutas que presenta la aplicación, así como las *Screen* que debe renderizar para cada una de estas rutas. El hecho de hacer uso de la *Layout*, como he mencionado anteriormente, es lo que permite que se rendericen los componentes de una manera u otra.

4.1.3. Resultados de la implementación

En este apartado voy a exponer los resultados del desarrollo del proyecto tanto por parte del *backend* como del *frontend* web. En el caso del *backend* mostraré una parte de la interfaz de *Swagger*. En el caso del *frontend* mostraré algunas de las interfaces con más relevancia de la plataforma. Cabe mencionar que algunas de las imágenes están recortadas ya que son excesivamente largas, es debido a esto que no voy a mostrar elementos repetidos, simplemente voy a reflejar los resultados.

Interfaz Swagger

En la figura 4.4 se puede observar algunos de los *endpoints* relacionados a las publicaciones y a los rankings. Esto se trata de una interfaz generada automáticamente por lo que yo no he intervenido directamente.

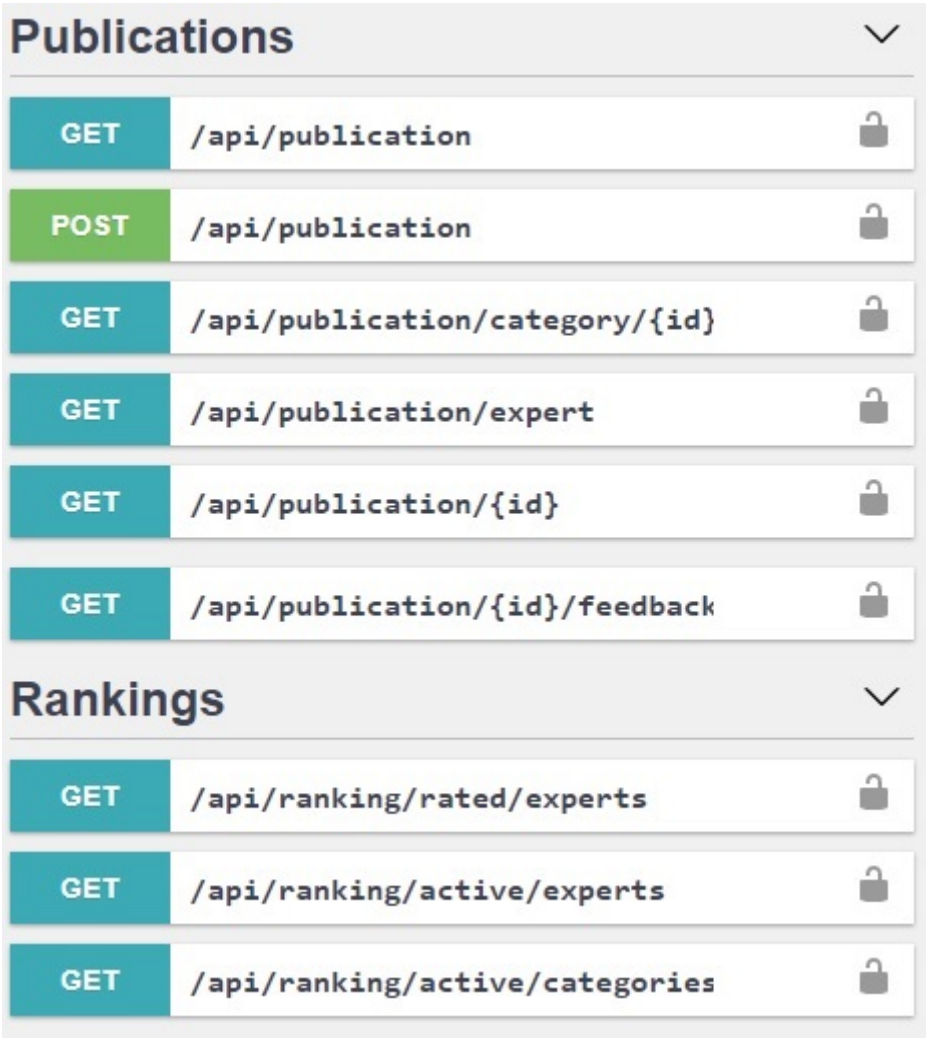

Figura 4.4: Interfaz *Swagger*.

Vista de login

En la figura 4.5 se puede observar el resultado de la vista del *login* del *frontend*, esta se trata de la vista por defecto cuando un usuario no esta autenticado en el sistema. Como puede observarse cumple con los criterios de la guía de estilo. También en ella no se muestra el *Header* como he expuesto anteriormente.

Vista de Publicaciones

En la figura 4.6 se puede observar el resultado de la vista del listado de publicaciones del *frontend*, en este caso la vista le pertenece a un aprendiz y por eso se pueden añadir publicaciones, si fuera un experto o un administrador no se podrían añadir. Como puede observarse cumple con los criterios de la guía de estilo. En este caso el *Header* sí que se muestra. Las lineas blancas se deben al recorte de la imagen.

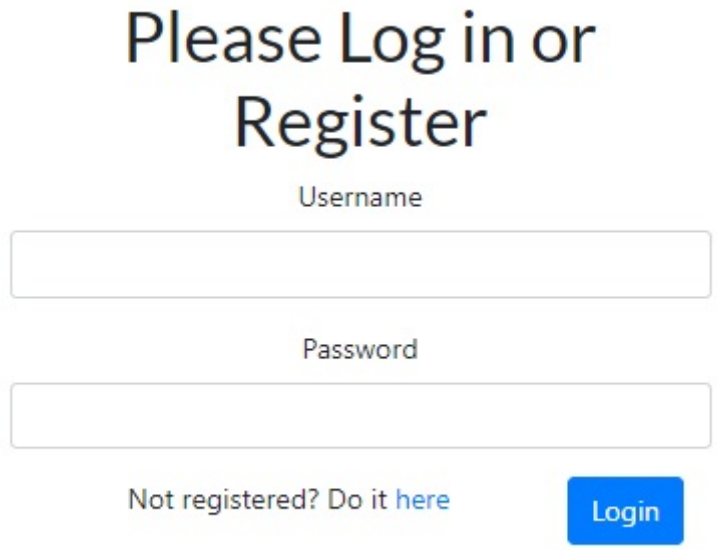

Figura 4.5: Vista del *Login*.

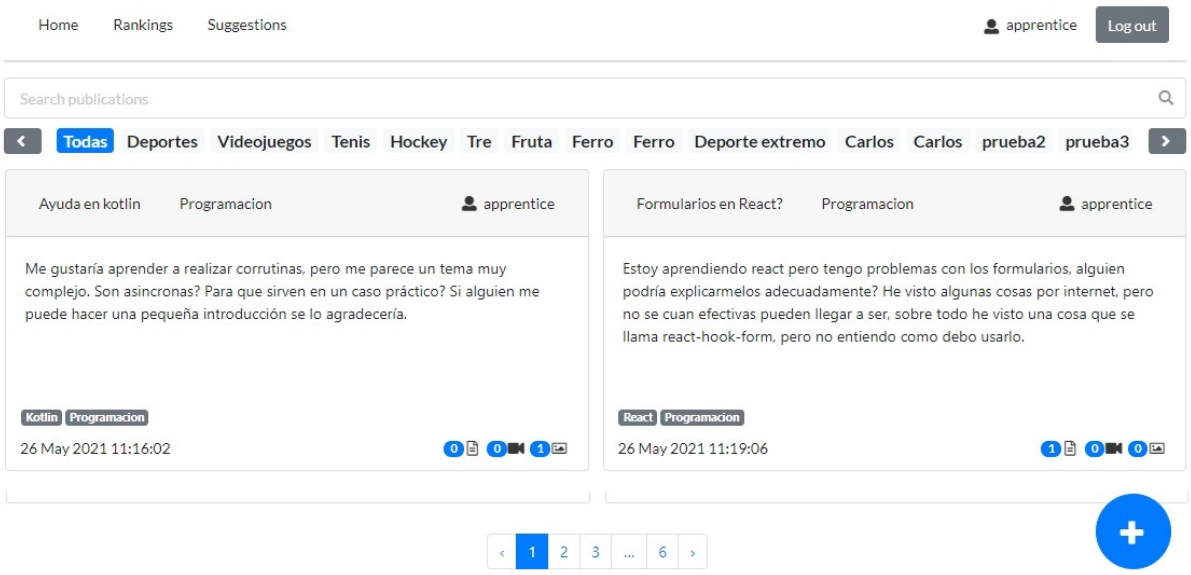

Figura 4.6: Vista de las Publicaciones.

Vista de Rankings

En la figura 4.7 se puede observar el resultado un fragmento de la vista del listado de los *rankings* del *frontend*, esta es una vista común a todos los tipos de usuarios. Como puede observarse cumple con los criterios de la guía de estilo. En este caso el *Header* sí que se muestra.

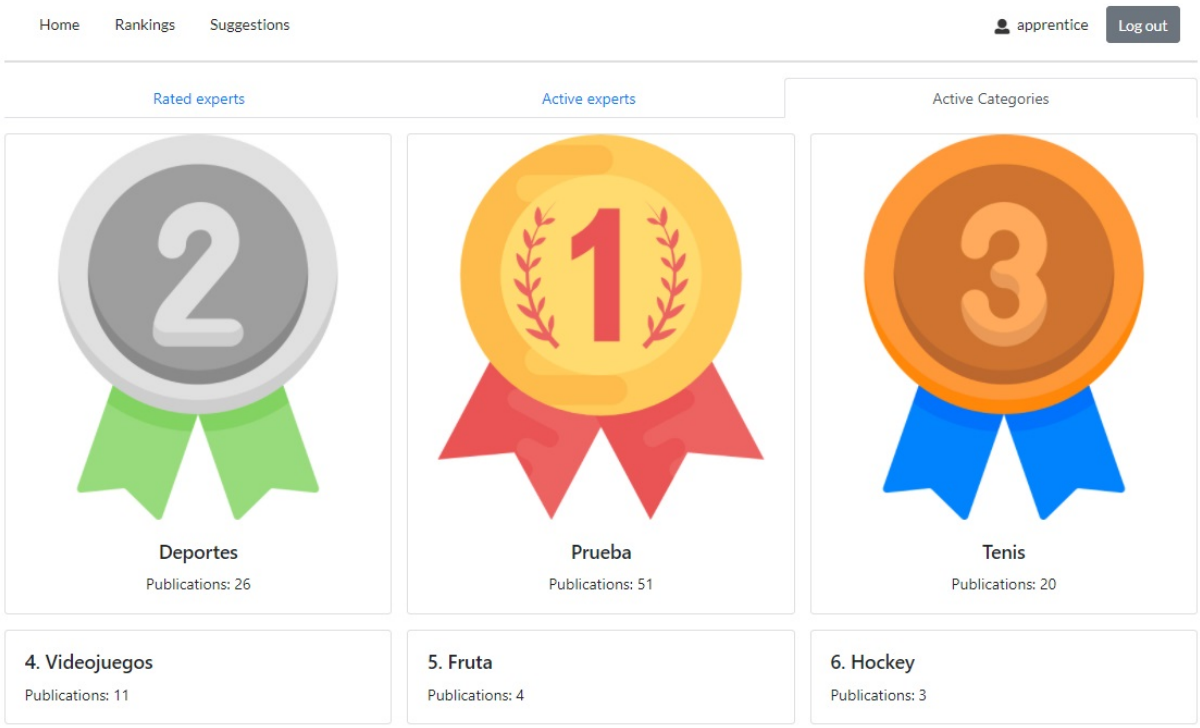

Figura 4.7: Vista de los *Rankings*.

Vista de Perfil

En la figura 4.8 se puede observar el resultado de la vista del perfil, de un aprendiz ene este caso, del *frontend*, esta vista seria igual para un experto, pero diferente en caso de ser un administrador. Como puede observarse cumple con los criterios de la guía de estilo. En este caso el *Header* sí que se muestra.

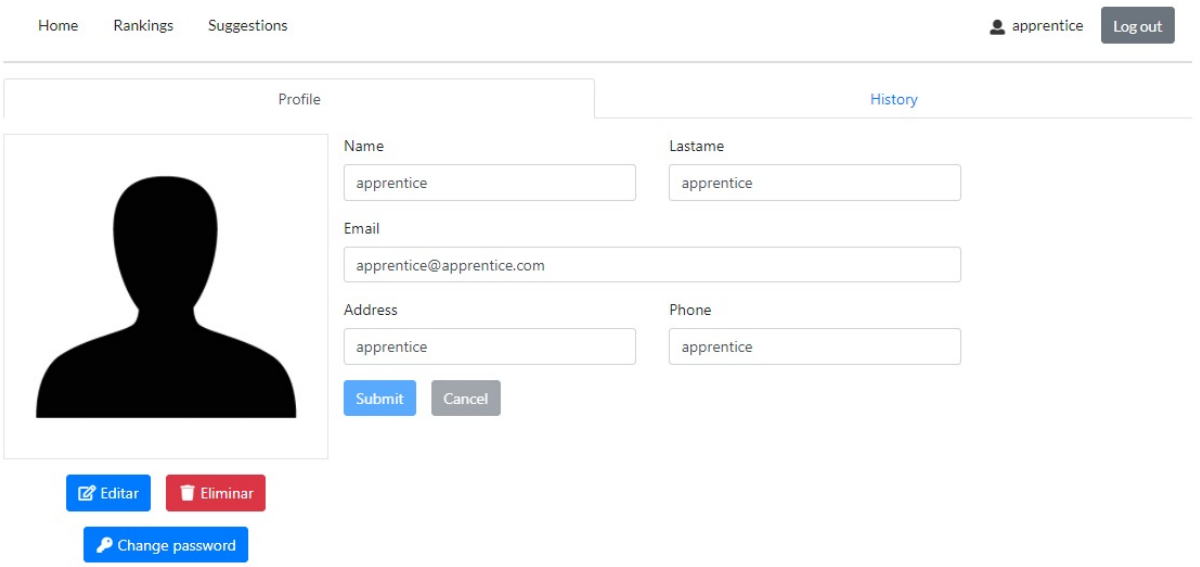

Figura 4.8: Vista del Perfil.

4.1.4. Problemas del desarrollo

En este apartado voy a exponer más en detalle los problemas que me han surgido a lo largo del desarrollo del proyecto y que expliqué brevemente en el apartado 2.5. Durante la implementación de este proyecto ha habido tres problemas clave que me han supuesto una complicación añadida a la hora de poder completar satisfactoriamente todos los objetivos propuestos. Estos son:

- Autenticación: En cuanto a la autenticación, mi problema surgió a la hora de establecer una autenticación que fuera adecuada y simple dentro de todas las posibilidades que existían dentro de *Symfony*. A la hora de desarrollar un sistema de autenticación, *Symfony* te permite crear un sistema totalmente propio, que requiera los parámetros que tu desees. En un principio pensé en hacer uso de esto, pero una vez que comencé a implementarlo me di cuenta que las opciones eran tantas y tan variadas que era sumamente complicado realizar una autenticación que funcionara correctamente.

  Ante esta dificultad decidí preguntar a mi mentor en la empresa como debía desarrollar este sistema de autenticación, para que pudiera darme pistas sobre como funcionaba realmente. El mentor me recomendó hacer uso de la autenticación mediante JWT (del inglés, *JSON Web Token*) y me recomendó que usará un sistema desarrollado por la comunidad que hacía uso de este sistema y que solo tendría que configurar unos pocos parámetros.

  Tras la reunión con el mentor comencé a realizar la autenticación como él me lo había recomendado y esta funcionaba correctamente y no costó mucho realizar la configuración inicial, por lo que finalmente puede completar el desarrollo de la autenticación del proyecto.

- Paginado: En cuanto al paginado, mi problema principal surgió porque este debía funcionar adecuadamente tanto para web como para una aplicación móvil. Originalmente hacerlo solo para web no supuso ningún problema, pero si fue complicado ajustarlo para que fuera adecuado en móvil.

  El principal problema en móvil era que el paginado empleaba listas infinitas, es decir, que tu podrías desplazar la lista hacía abajo hasta que no hubiera más elementos. Esto suponía un problema ya que si un usuario desplazaba la lista en el momento en el que otros usuarios añaden objetos a esa lista, de manera remota, en el momento en el que haga una petición para los siguientes elementos estos pueden estar repetidos.

  Para evitar que los elementos pudieran repetirse diseñé un sistema que funcionara tanto con números de página como con cursores. El *frontend* web haría uso de los números de páginas, mientras que el *frontend* móvil emplearía los cursores. Los cursores son simplemente los identificadores de las publicaciones, y se debe hacer uso del último cursor del listado del usuario para que al realizar una petición devuelva el listado de las publicaciones a partir de ese identificador y evitar que se repitan elementos.

- Notificaciones: En cuanto a las notificaciones el problema surgió tanto para el *backend* como el *frontend*. El problema del *backend* fue encolar debidamente las notificaciones, mientras que el problema del *frontend* fue el de suscribirme a recibir las notificaciones.

  Para poder mandar las notificaciones en el *backend* debía implementar un sistema de cola de notificaciones para que cada cierto tiempo, las notificaciones que estuvieran encoladas se mandaran, para evitar que estas se manden en cuanto fueran creadas, ya que podría llegar a suponer una carga excesiva para el servidor. El problema era que no podía compartir la cola entre los diferentes controladores y el servicio de notificaciones, debido a que cada uno

empleaba una copia local de la lista aunque implementara la inyección de dependencias. Para solucionarlo decidí que las notificaciones se almacenaran en la base de datos, para que la información pudiera compartirse entre todos los elementos implicados.

Para poder suscribirme a las notificaciones en el *frontend* tenía que obtener de alguna manera un token identificador que *OneSignal* supuestamente debía proporcionarme, pero no era capaz de obtenerlo ya que no sabía cual de las opciones debía emplear. Finalmente puede encontrar la opción correcta, que era realizar una petición concreta para que esta me devolviera el token identificador del dispositivo y pudiera emplearlo para recibir las notificaciones.

4.2. Verificación y validación

En esta sección voy a exponer todas las pruebas que he realizado para este proyecto. Estas pruebas engloban dos grandes tipos, primero tanto para el *backend* como el *frontend* he realizado una serie de pruebas de unidad. El siguiente gran grupo de pruebas que he realizado son pruebas de usuarios para comprobar que el *frontend* ha sido diseñado correctamente.

Las pruebas unitarias que he realizado en la parte del *backend* se han realizado con *Postman* en el que se ha hecho uso de los test que este puede implementar. Todas las pruebas en esta plataforma se han divido en una serie de colecciones que agrupan un conjunto de pruebas. Estas colecciones se encargan de probar un módulo concreto del *backend*. He realizado una colección de pruebas para cada gran controlador que he desarrollado en el *backend*.

Pongamos el ejemplo del controlador de publicaciones, para realizar esta colección de pruebas sigo el estándar de primero probar que si no estoy autenticado la petición debe de fallar, después compruebo que autenticado los diferentes tipos de peticiones que se pueden hacer funcionan correctamente. Una prueba es satisfactoria si esta devuelve el código http 200 (equivale a que todo a funcionado), la respuesta contiene los atributos que debe contener y esos atributos están formados correctamente. Esto pude observarse en el fragmento de código 4.7.

```
1  pm.test("Get publications", function () {
2      pm.response.to.have.status(200);
3      pm.response.to.not.be.error;
4      pm.response.to.be.withBody;
5      pm.response.to.have.jsonBody("publications");
6      pm.response.to.have.jsonBody("itemSize");
7      pm.response.to.have.jsonBody("leftSize");
8      pm.response.to.not.have.jsonBody("error");
9
10     var jsonData = JSON.parse(responseBody);
11     pm.expect(jsonData.publications[0]).to.exist;
12     pm.expect(jsonData.publications[0].id).to.exist;
13     if (jsonData.leftSize > 0)
14         pm.expect(jsonData.publications[jsonData.itemSize - 1]).to.exist;
15
16 });
```

Código Fuente 4.7: Prueba sobre publicaciones

Las pruebas unitarias que he realizado en la parte del *frontend* se han realizado con las pruebas nativas de *ReactJS*. Con estas pruebas mi objetivo era el de comprobar que los componentes se montaban adecuadamente, es decir, que dándole a los componentes los elementos que necesitaban para formarse, estos se formaban y contenían una serie de elementos que debían de contener. Además de esto también comprobaba que esos componentes se comportaban como debían, y que invocaban a las funciones que estos debían invocar cuando se interactuara con ellos.

Pongamos el ejemplo del componente de categoría favorita, lo primero que debía de comprobar es que el componente se montara adecuadamente, es decir que el contenedor de la vista contenía este componente. Tras esto comprobaba que el componente incluyera los elementos que lo formaban. Finalmente probaba que la interacción con ese componente funcionaba adecuadamente. Esto pude observarse en el fragmento de código 4.8.

```
//definir el contenedor

it("Favorite category is mounted and works", () => {
    //definir componente y funciones del mismo

    act(() => {
      render(card, container);
    });

    const icon = document.getElementById('star1')

    expect(contenedor!.textContent).toContain("TEST");
    expect(icon).toBeDefined()
    expect(icon?.className).toBe('star outline icon');

    act(() => {
      icon?.dispatchEvent(new MouseEvent("click", { bubbles: true }));
      icon?.dispatchEvent(new MouseEvent("click", { bubbles: true }));
    });

    expect(handleFav).toHaveBeenCalledTimes(2);
});
```

Código Fuente 4.8: Prueba sobre componente Categoría Favorita

Finalmente, en cuanto a las pruebas de usuario, estas se tratan de hacer que usuarios potenciales de la plataforma, en mi caso la web, la prueben y la evalúen de manera personal. Lo que se va a realizar con estas pruebas es que una serie de usuarios seleccionados realizarán diferentes tareas sobre la web, tras realizar estas tareas deberán completar un informe sobre que les ha parecido la web y los resultados de esos informes nos darán la información de si la web es adecuada y esta bien diseñada, o si en caso contrario no lo está.

En condiciones normales se requerirían un mayor número de personas, pero en mi caso he realizado simplemente 4 pruebas de usuarios. Estas pruebas se van a dividir para los dos tipos de usuarios principales que componen el sistema, dos de estos usuarios serán tratados como expertos mientras que los otros dos serán tratados como aprendices. Todas las personas que van a participar en estas pruebas son personas ajenas al proyecto, y forman parte de mi circulo de

amistad, ya que para esto no se requieren de expertos en el desarrollo de interfaces web.

Ambas grupos de usuarios deberán realizar una serie de tareas sobre la web, algunas de ellas serán comunes a ambos usuarios, pero otras serán exclusivas de esos tipos de usuarios. Estas pruebas se realizarán de manera independiente a cada usuario, para que todo el mundo llegue como un novato pro primera vez a la web. Las tareas que realizarán son las siguientes:

- Entrar a la aplicación.
- Listar las publicaciones.
- Realizar una publicación (solo aprendices).
- Acceder a una publicación concreta.
- Consultar un archivo.
- Realizar una retroalimentación (solo expertos).
- Consultar rankings.
- Enviar una sugerencia.
- Reportar una publicación.
- Editar el perfil.
- Consultar historial.
- Valorar una retroalimentación (solo aprendices).
- Añadir categoría favorita (solo expertos).

Estas tareas le serán dadas previamente a cada persona que realizara las pruebas. Una vez se les haya proporcionado el listado de tareas se les dejará en la dirección base de la web y a partir de ese momento deberán comenzar a realizar las tareas una tras otra. En caso de que no puedan completar una tarea del listado, deberán saltarla y proseguir con la siguiente. En todo momento serán monitorizados para comprobar sus expresiones corporales y faciales y tener una primera idea de si les esta resultando sofocante o por el contrario se encuentran relajados.

Una vez hayan completado la prueba, independientemente del número de tareas que no hayan podido realizar, deberán completar el formulario QUIS (del inglés, *Questionnaire for User Interface Satisfaction*)[20] que se les proporcionará, en el contestarán una serie de preguntas relevantes a la interfaz web, después deberán indicar los aspectos más positivos y más negativos sobre la web para finalmente rellenar una serie de datos que indicarán su nivel de conocimiento sobre este tipo de plataformas.

Para mostrar los resultados, los dividiré en 5 grandes grupos, que corresponden a los diferentes apartados generales que están presentes en el formulario QUIS y para cada grupo mostraré el valor medio de todos los aspectos que intervienen en él, por ejemplo si el apartado general presenta 6 características haré uso de la media de esos valores. Mostraré los resultados individuales de cada uno, y luego mostraré un resultado general para las 4 pruebas. Estos resultados pueden observarse en el cuadro 4.1

	Resultados					
	Overall	**Screen**	**Terminology**	**Learning**	**Capabilities**	**Usuario**
A1	6,00	7,00	7,67	8,00	6,80	7,09
A2	8,00	7,66	7,15	9,00	8,60	8,08
E1	6,50	7,33	7,17	8,00	6,40	7,08
E2	7,66	9,00	8,17	9,00	8,40	8,45
Grupo	7,04	7,75	7,54	8,50	7,55	7,68

Cuadro 4.1: Resultados pruebas de usuario.

En el cuadro, los usuarios que han probado como aprendices son los que se marcan como A1 y A2, mientras que los que han probado como expertos son los que se marcan como E1 y E2. La columna usuario representa la nota media que ese usuario ha dado a todos los grupos de preguntas. La fila Grupo representa la nota media de todos los usuarios para ese grupo, o para la nota media de todos los grupos.

Por lo que se puede observar en el cuadro, el apartado en el que menos nota ha sacado la web es en el apartado general, mientras que en el que más nota ha sacado es en el de aprendizaje. También se puede observar que la nota media final por parte de todos los usuarios que han participado en la prueba es de 7,68.

Lo que podemos observar con esto es que de manera general la web no es completamente adecuada, pero a pesar de esto es bastante aceptable pues no supone un alto aprendizaje. También podemos decir que las pantallas y las secuencias de las mismas son bastante acertadas. Se puede apreciar que también es bastante adecuada en cuanto a la terminología. Por último, también es bastante completa en cuanto a las capacidades de la misma.

En definitiva, podemos concluir con todos estos datos es que la web, si bien es cierto presenta elementos que la vuelven un poco rígida o incompleta, pero hay otros aspectos en los que triunfa especialmente, por lo que podemos considerar los resultados de estas pruebas como satisfactorios.

Los informes completos, que incluyen todas las notas individuales de todos los grupos, el listado de los aspectos más negativos y positivos, y la información sobre la persona que ha realizado la prueba se encuentran en el anexo B.

Capítulo 5

Conclusiones

Primero, a nivel personal, esta estancia en prácticas y este proyecto han sido muy enriquecedores en diferentes ámbitos, tanto el formativo, como el profesional, como el personal.

A nivel formativo he sido capaz de aprender tecnologías que no había empleado anteriormente, tanto el *backend* con PHP y *Symfony*, como el *frontend* con *ReactJS*. La curva de aprendizaje con estos ha sido muy satisfactoria, a pesar de los problemas que me surgieron, ha sido una muy bonita experiencia el ser capaz de aprender a usarlos desde cero.

A nivel laboral he podido aprender como funciona una empresa con las características de 480 por dentro. He aprendido a evaluar como este tipo de empresas realizan y tratan los proyectos en los que trabajan y como los consiguen sacar hacia delante. He conseguido aprender como manejar los problemas que me surgen de imprevisto y ser capaz de resolverlos.

A nivel personal, he sido capaz de conocer y tratar con mucha gente que me ha ayudado a lo largo del desarrollo del proyecto. Estas personas me han ayudado a completar el proyecto de manera satisfactoria y me han sabido enseñar cosas nuevas sobre las bases que tenía, impulsándome a mejorar esas bases y reforzarlas para hacer un mejor trabajo y sentirme totalmente satisfecho con lo que he logrado.

A nivel personal también, el hecho de realizar este proyecto de manera conjunta me ha aportado mucho a nivel de trabajo en equipo, hemos tenido que aprender a coordinar nuestro trabajo adecuadamente, hemos recibido ayuda el uno del otro y hemos conseguido fortalecer la amistad que teníamos antes.

En definitiva, estas prácticas y este proyecto, me han ayudado a prepararme para mis siguientes objetivos profesionales. Me han ayudado a darme el empujón que necesitaba para realmente conocer como se debe desarrollar un proyecto de este calibre. Me han permitido aprender cosas que deseaba aprender y me han permitido abrirme a las nuevas cosas que quiero aprender.

Ahora en cuanto a nivel técnico del proyecto, todos los objetivos que se habían planteado originalmente para este proyecto, han sido completados satisfactoriamente. También hemos podido completar los objetivos que se fueron añadiendo conforme se iba avanzando en el proyecto.

Es por esto, que en definitiva, podemos concluir que el desarrollo de este proyecto ha sido completado al completo de manera satisfactoria, y cumpliendo todos los requisitos expuestos al comienzo de su desarrollo. Podemos decir también que la plataforma cumple con los objetivos que dieron paso a la necesidad de realizarla.

Bibliografía

[1] 480. Cuatroochenta. https://cuatroochenta.com, 2021. Consulta: 19 de mayo de 2021.

[2] Encarna Abellan. *Scrum*: qué es y cómo funciona esta metodología. https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html, 2020. Consulta: 5 de marzo de 2020.

[3] Mehdi Achour, Fiedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, and Jakub Vrana. Manual de php. https://www.php.net/manual/es/index.php, 2021. Consulta: 8 de mayo de 2021.

[4] José María Aguilar. Qué es la inyección de dependencias y cómo funciona. https://www.campusmvp.es/recursos/post/que-es-la-inyeccion-de-dependencias-y-como-funciona.aspx, 2020. Consulta: 25 de agosto de 2020.

[5] Axios. Axios getting started. https://axios-http.com/docs/intro, 2021. Consulta: 4 de enero de 2021.

[6] Bootstrap. Bootstrap documentation. https://getbootstrap.com/docs/5.0/getting-started/introduction/, 2021. Consulta: 5 de mayo de 2021.

[7] Doctrine. Doctrine. https://www.doctrine-project.org, 2021. Consulta: 24 de mayo de 2021.

[8] Facebook. React. https://es.reactjs.org, 2021.

[9] Git. Git documentation. https://git-scm.com/docs, 2021.

[10] Node JS. Node documentación. https://nodejs.org/en/docs/, 2021. Consulta: 19 de mayo de 2021.

[11] Jack Lukic. Semantic ui documentation. https://semantic-ui.com/introduction/getting-started.html, 2018. Consulta: 13 de octubre de 2018.

[12] Theo Mandel. Golden rules of user interface design. https://theomandel.com/resources/golden-rules-of-user-interface-design/, 1997. Consulta: 19 de mayo de 2021.

[13] BBVA API Market. Api rest: qué es y cuáles son sus ventajas en el desarrollo de proyectos. https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/, 2016. Consulta: 23 de marzo de 2016.

[14] Felipe Medina. Refactorización y mejora de código. https://medium.com/@XFelipeM/refactorizacion-de-codigo-like-a-champion-1-a9c9c95d704f, 2017. Consulta: 11 de noviembre de 2017.

[15] Scrum Mexico. Escribiendo historias de usuario. https://scrum.mx/informate/historias-de-usuario, 2018. Consulta: 2 de agosto de 2018.

[16] Microsoft. Typescript documentation. https://getbootstrap.com/docs/5.0/getting-started/introduction/, 2021. Consulta: 10 de abril de 2021.

[17] Darrel Millera, Jeremy Whitlock, Marsh Gardiner, Mike Ralphson, Ron Ratovsky, and Uri Sarid. Openapi specification. https://spec.openapis.org/oas/v3.1.0, 2021. Consulta: 15 de febrero de 2021.

[18] Mozilla. Http tutoriales. https://developer.mozilla.org/es/docs/Web/HTTP, 2021. Consulta: 23 de mayo de 2021.

[19] Mozilla. Javascript. https://developer.mozilla.org/es/docs/Web/JavaScript, 2021. Consulta: 27 de mayo de 2021.

[20] University of Maryland. Quis: The questionnaire for user interaction satisfaction. https://www.cs.umd.edu/hcil/quis/, 1999.

[21] OneSignal. Onesignal getting started. https://documentation.onesignal.com/docs, 2021. Consulta: 20 de abril de 2021.

[22] Dan Radigan. Puntos de historia y estimación. https://www.atlassian.com/es/agile/project-management/estimation, 2021. Consulta: 15 de marzo de 2021.

[23] Uriel Ruelas. ¿qué es la serialización o *marshalling*? https://codingornot.com/que-es-la-serializacion-o-marshalling, 2017. Consulta: 19 de mayo de 2017.

[24] SMARTBEAR. Api endpoints - what are they? why do they matter? https://smartbear.com/learn/performance-monitoring/api-endpoints/, 2021. Consulta: 24 de mayo de 2021.

[25] Swagger. Swagger. https://swagger.io, 2021. Consulta: 24 de mayo de 2021.

[26] Vanessa Rosselló Villán. Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/, 2019. Consulta: 15 de marzo de 2019.

[27] Dealer World. Modalidad '*Cloud*': una solución ideal para implementar soluciones digitales. https://www.dealerworld.es/tendencias/modalidad-cloud-una-solucion-ideal-para-implementar-soluciones-digitales, 2020. Consulta: 2 de abril de 2020.

Anexo A

Tablas de especificación de los casos de uso

Especificación del caso de uso	
Identificador	CU03
Nombre	Crear categorías.
Versión	V01
Autor	Alex Martínez Martínez
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los administrados puedan crear categorías.
Alcance	El sistema ha de permitir que los administrados puedan crear categorías, tanto categorías base como subcategorías.
Nivel	Tarea principal
Actor principal	Administrador
Actores secundarios	N/A
Relaciones	CU04, CU08
Precondición	El administrador ha de estar registrado como administrador en el sistema.
Condición fin con éxito	La categoría será creada y añadida a la lista de categorías.
Condición fin con fracaso	La categoría no será creada y no será añadida a la lista de categorías.
Trigger	Se quiere crear una nueva categoría para que haya más opciones al crear publicaciones.
Secuencia normal	**Acción**
1	El administrador se autentifica en la aplicación.
2	El administrador accede a la vista de crear categorías.
3	El administrador rellena toda la información necesaria sobre la categoría.
4	El administrador crea la categoría.
5	El sistema almacena la categoría y notifica al administrador que esta ha sido creada satisfactoriamente.
Excepción en 5 - Categoría padre no encontrada	**Excepción**
1	El sistema no almacena la categoría y notifica el error.
2	El administrador rellena toda la información necesaria sobre la categoría.
3	El administrador crea la categoría.
4	El sistema almacena la categoría y notifica al administrador que esta ha sido creada satisfactoriamente.
Excepción en 5 - Usuario no válido	**Excepción**
1	El sistema no almacena la categoría y notifica que el usuario no es válido.
2	El administrador rellena toda la información necesaria sobre la categoría.
3	El administrador crea la categoría.
4	El sistema almacena la categoría y notifica al administrador que esta ha sido creada satisfactoriamente.
Excepción en 4 - Cancelar operación	**Excepción**
1	El administrador cancela la operación.
Frecuencia esperada	1 vez cada dos semanas
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.1: Especificación del caso de uso CU03.

Especificación del caso de uso	
Identificador	CU04
Nombre	Añadir o eliminar categoría favorita.
Versión	V01
Autor	Carlos Mora Hernández
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los expertos puedan marcar o desmarcar categorías como favoritas.
Alcance	El sistema ha de permitir que los expertos puedan marcar (en caso de que no esté marcada) o desmarcar (en caso de que este marcada) categorías como favoritas.
Nivel	Tarea principal
Actor principal	Experto
Actores secundarios	N/A
Relaciones	CU03, CU08
Precondición	El experto ha de estar registrado en el sistema.
Condición fin con éxito	La categoría se marca o desmarca como favorita.
Condición fin con fracaso	La categoría no se marca o no se desmarca como favorita.
Trigger	Se quiere añadir o eliminar una categoría como favorita para acceder o dejar de acceder a ella.
Secuencia normal	**Acción**
1	El experto se autentifica en la aplicación.
2	El experto accede a la vista de categorías favorita.
3.1	El experto marca la categoría como favorita.
3.2	El sistema almacena la categoría como favorita y notifica al experto que esta ha sido añadida.
4.1	El experto desmarca la categoría como favorita.
4.2	El sistema almacena la categoría como no favorita y notifica al experto que esta ha sido eliminada.
Excepción en 3.2 - Categoría no encontrada	**Excepción**
1	El sistema no almacena la categoría como favorita y notifica el error.
2.1	El experto marca la categoría como favorita.
2.2	El sistema almacena la categoría como favorita y notifica al experto que esta ha sido añadida.
Excepción en 3.2 - Categoría ya es favorita	**Excepción**
1	El sistema no almacena la categoría como favorita y notifica el error.
2.1	El experto marca la categoría como favorita.
2.2	El sistema almacena la categoría como favorita y notifica al experto que esta ha sido añadida.
Excepción en 4.2 - Categoría no encontrada	**Excepción**
1	El sistema no almacena la categoría como no favorita y notifica el error.
2.1	El experto desmarca la categoría como favorita.
2.2	El sistema almacena la categoría como no favorita y notifica al experto que esta ha sido añadida.
Excepción en 4.2 - Categoría no es favorita	**Excepción**
1	El sistema no almacena la categoría como favorita y notifica el error.
2.1	El experto marca la categoría como favorita.
2.2	El sistema almacena la categoría como favorita y notifica al experto que esta ha sido añadida.
Frecuencia esperada	50 veces al día
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.2: Especificación del caso de uso CU04.

Especificación del caso de uso	
Identificador	CU06
Nombre	Consultar historial.
Versión	V01
Autor	Carlos Mora Hernández
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que tanto expertos como aprendices puedan consultar su historial.
Alcance	El sistema ha de permitir que tanto expertos como aprendices puedan consultar su historial mostrando publicaciones y feedbacks realizados, además de valoraciones y respuestas obtenidas
Nivel	Tarea principal.
Actor principal	Aprendiz, Experto
Actores secundarios	N/A
Relaciones	N/A
Precondición	El usuario ha de estar registrado en el sistema.
Condición fin con éxito	El historial se lista y se puede consultar.
Condición fin con fracaso	El historial no se lista y no se puede consultar.
Trigger	El usuario quiere ver todo lo que ha acontecido durante su estancia en la aplicación.
Secuencia normal	**Acción**
1	El usuario se autentifica en la aplicación.
2	El usuario accede a la vista del historial.
3	El sistema lista el historial del usuario.
4	El usuario consulta su historial.
5	El sistema devuelve la información sobre el historial.
Excepción en 5 - Error de servidor	**Excepción**
1	El sistema no lista el historial del usuario y muestra mensaje de lista vacía.
2	El usuario accede a la vista del historial.
3	El sistema lista el historial del usuario.
4	El usuario consulta su historial.
5	El sistema devuelve la información sobre el historial.
Frecuencia esperada	200 veces al día
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.3: Especificación del caso de uso CU06.

Especificación del caso de uso	
Identificador	CU07
Nombre	Consultar, modificar y eliminar datos del perfil.
Versión	V01
Autor	Alex Martínez Martínez
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los usuarios puedan consultar, modificar y eliminar datos de su perfil.
Alcance	El sistema ha de permitir que los usuarios puedan consultar, modificar y eliminar datos de su perfil.
Nivel	Tarea principal
Actor principal	Aprendiz, Experto
Actores secundarios	Adminsitrador
Relaciones	N/A
Precondición	El usuario ha de estar registrado en el sistema.
Condición fin con éxito	El sistema devuelve los datos del perfil, y estos pueden modificarse y/o eliminarse.
Condición fin con fracaso	El sistema no devuelve los datos del perfil, y estos pueden modificarse y/o eliminarse.
Trigger	El usuario quiere ver sus datos personales y editarlos o eliminarlos.
Secuencia normal	**Acción**
1	El usuario se autentifica en la aplicación.
2	El usuario accede a la vista del perfil.
3	El sistema devuelve los datos del perfil.
4.1	El usuario modifica los datos de su perfil.
4.2	El sistema guarda los nuevos datos del perfil.
5.1	El usuario elimina los datos de su perfil.
5.2	El sistema elimina los datos del perfil y el usuario deja de formar parte de la aplicación.
Excepción en 4.1 y 5.1 - Cancelar la operación	**Excepción**
1	El usuario cancela la operación.
Excepción en 4.2 - Usuario ya existe	**Excepción**
1	El sistema no guarda los nuevos datos del perfil y notifica el error.
2.1	El usuario modifica los datos de su perfil.
2.2	El sistema guarda los nuevos datos del perfil.
Excepción en 5.2 - Contraseña incorrecta	**Excepción**
1	El sistema no elimina los datos del perfil y notifica el error.
2.1	El usuario elimina los datos de su perfil.
2.2	El sistema elimina los datos del perfil y el usuario deja de formar parte de la aplicación.
Frecuencia esperada	50 veces al día
Importancia	Necesaria
Prioridad	Medio plazo
Comentarios	N/A

Cuadro A.4: Especificación del caso de uso CU07.

Especificación del caso de uso	
Identificador	CU08
Nombre	Listar categorías.
Versión	V01
Autor	Carlos Mora Hernández
Fuente	Sergio Aguado González (Supervisor)
Descripción	El sistema ha de permitir que los usuarios puedan listar las categorías.
Alcance	El sistema ha de permitir que los usuarios puedan listar las categorías, tantolas categorías base como las subcategorías.
Nivel	Tarea principal
Actor principal	Aprendiz, Experto
Actores secundarios	Administrador
Relaciones	CU03
Precondición	El usuario ha de estar registrado en el sistema.
Condición fin con éxito	El sistema devuelve el listado de las categorías, y estas pueden consultarse.
Condición fin con fracaso	El sistema no devuelve el listado de las categorías, y estas no pueden consultarse.
Trigger	El usuario quiere ver un listado con todas las categorías existentes.
Secuencia normal	**Acción**
1	El usuario se autentifica en la aplicación.
2	El usuario accede a la vista del listado de categorías.
3	El sistema lista las categorías.
4	El usuario consulta las categorías.
Excepción en 3 - Error de servidor	**Excepción**
1	El sistema no devuelve el listado de las categorías y muestra un mensaje de lista vacía.
2	El usuario accede a la vista del listado de categorías.
3	El sistema lista las categorías.
4	El usuario consulta las categorías.
Frecuencia esperada	50 veces al día
Importancia	Necesaria
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.5: Especificación del caso de uso CU08.

Especificación casos de uso	
Identificador	CU09
Nombre	Valorar feedback
Versión	1.0
Autores	Alex Martínez Martínez
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El aprendiz que a recibido un feedback debe poder valorarlo para poder retroalimentar al experto
Alcance	El sistema permite que se realicen valoraciones solo si son autores de la publicación asociada y la nota variará del 1 al 10.
Nivel	Subtarea
Actor principal	Aprendiz
Actores secundarios	Administrador
Relaciones	N/A
Precondición	El aprendiz debe estar autenticado en el sistema.
Condición fin con éxito	Se envía la valoración al experto.
Condición fin con fracaso	El experto no recibe una valoración
Trigger	El experto realiza un feedback sobre una publicación.
Secuencia normal	**Acción**
1	El aprendiz se autentica en el sistema.
2	El aprendiz entra a la pantalla de un feedback recibido sobre una publicación.
3	El aprendiz valora dicho feedback.
4	Se envía la valoración al servidor.
Excepción en 3 - Valoración fuera de rango	**Excepción**
1	El aprendiz valora dicho feedback con una nota fuera de rango.
2	El sistema notifica al aprendiz y no envía la valoración
Frecuencia esperada	10 al día
Importancia	Secundaria
Prioridad	Medio plazo
Comentarios	N/A

Cuadro A.6: Especificación del caso de uso CU09

Especificación casos de uso	
Identificador	CU10
Nombre	Crear sugerencias
Versión	1.0
Autores	Carlos Mora Hernández
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El experto o aprendiz deben poder enviar a los administradores sugerencias de mejora y nuevas categorias que se pueden añadir al sistema.
Alcance	El sistema permite sugerencias de mejora y de nuevas categorias, tras ello se envian a los administradores, los cuales las reciben en un buzón.
Nivel	Subtarea
Actor principal	Aprendiz y experto.
Actores secundarios	Administrador
Relaciones	N/A
Precondición	El experto o aprendiz debe estar autenticado en el sistema.
Condición fin con éxito	El administrador recibe dichas sugerencias.
Condición fin con fracaso	El administrador no recibe dichas sugerencias.
Trigger	El experto o aprendiz encuentran una posible mejora en el sistema o una categoría que no existe.
Secuencia normal	**Acción**
1	El experto o aprendiz se autentifica en el sistema.
2	El experto o aprendiz entran en la pantalla de sugerencias.
3	El experto o aprendiz especifican la mejora o la categoría nueva que se podría introducir.
4	Se envía la sugerencia al servidor.
5	El administrador recibe dicha sugerencia y toma acciones para aplicarla.
Excepción en 3 - Sugerencia vacía	**Excepción**
1	El experto o aprendiz envían una sugerencia vacía
2	El sistema notifica al experto o aprendiz indicándole que especifique un mensaje.
Frecuencia esperada	1 a la semana
Importancia	Secundaria
Prioridad	Medio plazo
Comentarios	N/A

Cuadro A.7: Especificación del caso de uso CU10

Especificación casos de uso	
Identificador	CU11
Nombre	Repostar publicación
Versión	1.0
Autores	Alex Martínez Martínez
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El usuario debe poder reportar alguna publicación que le parezca ofensiva o inadecuada.
Alcance	El sistema debe prestar la opción al usuario de reportar una publicación por Spam o por contenido inadecuado.
Nivel	Subtarea
Actor principal	Aprendiz y experto.
Actores secundarios	N/A
Relaciones	CU02
Precondición	El usuario debe estar autenticado en el sistema.
Condición fin con éxito	Llega al servidor el reporte de la publicación.
Condición fin con fracaso	Llega al servidor el reporte de la publicación.
Trigger	El usuario encuentra una publicación inadecuada.
Secuencia normal	**Acción**
1	El usuario se autentifica en el sistema.
2	El usuario encuentra una publicación inadecuada.
3	El usuario reporta la publicación encontrada.
4	Se envía el reporte al servidor.
5	El sistema notifica al usuario de que la publicación ha sido reportada.
Excepción en 5 - Fallo de comunicación	**Excepción**
1	La petición falla y se notifica al usuario del problema que ha surgido.
Frecuencia esperada	2 al dia
Importancia	Secundaria
Prioridad	Medio plazo
Comentarios	N/A

Cuadro A.8: Especificación del caso de uso CU11

Especificación casos de uso	
Identificador	CU12
Nombre	Buscar publicaciones
Versión	1.0
Autores	Carlos Mora Hernández
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El usuario debe poder buscar las publicaciones que le interesen mediante un buscador.
Alcance	El sistema, a partir de los datos del usuario, debe poder filtrar publicaciones a partir de su nombre, el autor o los tags que contiene.
Nivel	Subtarea
Actor principal	Aprendiz y experto.
Actores secundarios	N/A
Relaciones	CU02
Precondición	El usuario debe estar autenticado en el sistema.
Condición fin con éxito	El usuario encuentra la publicación que buscaba.
Condición fin con fracaso	El usuario no encuentra la publicación que buscaba.
Trigger	El usuario accede a la vista de filtraje del sistema.
Secuencia normal	**Acción**
1	El usuario se autentifica en el sistema.
2	El usuario accede a la vista de filtraje del sistema.
3	El usuario introduce el criterio de búsqueda de una publicación.
4	El sistema busca dicha publicación.
5	El sistema muestra la/s publicación/es que coincide con ese criterio.
Excepción en 5 - Sin resultados	**Excepción**
1	El sistema muestra un mensaje especificando que no se han encontrado publicaciones que cumplan con los criterios descritos.
Frecuencia esperada	100 al dia
Importancia	Secundaria
Prioridad	Medio plazo
Comentarios	N/A

Cuadro A.9: Especificación del caso de uso CU12

Especificación casos de uso	
Identificador	CU13
Nombre	Autenticarse en la aplicación
Versión	1.0
Autores	Alex Martínez Martínez
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El usuario debe poder autenticarse en la aplicación para poder acceder a todas sus funcionalidades.
Alcance	El sistema permitirá la autenticación de usuarios a partir de un nombre de usuario y una contraseña, ambos especificados durante el registro.
Nivel	Tarea principal
Actor principal	Aprendiz, experto y administrador
Actores secundarios	N/A
Relaciones	CU15
Precondición	El usuario debe estar registrado en el sistema.
Condición fin con éxito	El usuario se autentifica en el sistema y a todas sus funcionalidades.
Condición fin con fracaso	El usuario no se autentifica en el sistema.
Trigger	Un usuario entra en la aplicación.
Secuencia normal	**Acción**
1	El usuario entra en la aplicación
2	El usuario introduce el nombre de usuario y su contraseña.
3	El sistema valida que tanto el usuario como la contraseña son correctos.
4	El usuario accede al sistema y a sus funcionalidades.
Excepción en 3 - Contraseña incorrecta	**Excepción**
1	El sistema valida que el usuario y la contraseña son incorrectos.
2	El sistema notifica al usuario que alguno de los datos es incorrecto.
Frecuencia esperada	300 al dia
Importancia	Vital
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.10: Especificación del caso de uso CU13

Especificación casos de uso	
Identificador	CU14
Nombre	Listar rankings
Versión	1.0
Autores	Alex Martínez Martínez
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El usuario debe poder ver los diferentes rankings que existen en el sistema.
Alcance	El sistema debe prestar a los usuarios tres rankings, uno de categorías más activas, otro de expertos más activos y el último de expertos mejor valorados.
Nivel	Tarea principal
Actor principal	Aprendiz y experto
Actores secundarios	N/A
Relaciones	N/A
Precondición	El usuario debe estar autenticado en el sistema.
Condición fin con éxito	La lista de rankings es visualizada por el usuario
Condición fin con fracaso	El sistema no muestra la lista al usuario.
Trigger	Un usuario entra en la pantalla de rankings.
Secuencia normal	**Acción**
1	El usuario se autentifica en el sistema.
2	El entra en la pantalla de rankings.
3	El usuario visualiza uno de los rankings y tiene la opción de ver el resto.
Excepción en 3 - No se muestra la lista	**Excepción**
1	El usuario no visualiza los rankings debido a que está vacía, se le muestra un mensaje notificándoselo.
Frecuencia esperada	200 al dia
Importancia	Vital
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.11: Especificación del caso de uso CU14

Especificación casos de uso	
Identificador	CU15
Nombre	Registrarse en la aplicación
Versión	1.0
Autores	Carlos Mora Hernández
Fuentes	Sergio Aguado González (Supervisor)
Descripción	El usuario debe poder registrarse en la aplicación introduciendo sus datos personales y eligiendo el rol que quiere realizar.
Alcance	El sistema debe permitir elegir el rol al usuario, así como que se puedan introducir los datos personales, comprobar que la contraseña cumple con los mínimos y que el nombre de usuario es único.
Nivel	Tarea principal
Actor principal	Aprendiz y experto
Actores secundarios	N/A
Relaciones	N/A
Precondición	El usuario debe entrar en el sistema.
Condición fin con éxito	El usuario esta registrado en el sistema y puede acceder a él.
Condición fin con fracaso	El usuario no se ha registrado en el sistema.
Trigger	Un usuario entra en la pantalla de registro.
Secuencia normal	**Acción**
1	El usuario entra en la pantalla de registro.
2	El usuario introduce sus datos en el sistema.
3	El sistema valida esos datos.
4	El sistema notifica al usuario indicándole que esta registrado en el sistema.
Excepción en 2 - Nombre de usuario repetido	**Excepción**
1	El sistema valida los datos.
2	El sistema notifica al usuario indicando que el nombre de usuario introducido ya está en uso.
3.1	El usuario cambia el nombre de usuario por otro.
3.2	El sistema valida esos datos.
3.3	El sistema notifica al usuario indicándole que esta registrado en el sistema.
Excepción en 2 - Contraseñas no coinciden	**Excepción**
1	El sistema valida los datos.
2	El sistema notifica al usuario indicando que la contraseña del primer campo no coincide con la del segundo campo.
3.1	El usuario cambia dichos campos para que sean iguales sus datos.
3.2	El sistema valida esos datos.
3.3	El sistema notifica al usuario indicándole que esta registrado en el sistema.
Frecuencia esperada	10 al dia
Importancia	Vital
Prioridad	Corto plazo
Comentarios	N/A

Cuadro A.12: Especificación del caso de uso CU15

Especificación casos de uso	
Identificador	CU16
Nombre	Recibir notificaciones
Versión	1.0
Autores	Carlos Mora Hernández
Fuentes	Sergio Aguado González(Supervisor)
Descripción	El sistema ha enviar una notificación de tipo push cuando alguien reaccione a una publicación o a un feedback del usuario.
Alcance	El sistema debe detectar cuando alguien reacciona a alguna publicación o feedback del usuario y debe notificar mediante un texto breve al usuario implicado.
Nivel	Subtarea
Actor principal	Aprendiz y Experto
Actores secundarios	N/A
Relaciones	CU09, CU05
Precondición	El usuario debe estar autenticado en el sistema como experto o aprendiz
Condición fin con éxito	El autor de la publicación o feedback recibe una notificación.
Condición fin con fracaso	El sistema muestra un mensaje de error al usuario que ha valorado o realizado un feedback comunicando que no se ha podido enviar la notificación push
Trigger	Un usuario realiza un feedback o una valoración
Secuencia normal	**Acción**
1	El usuario se autentica en el sistema como aprendiz.
2	El usuario busca uno de los feedbacks recibidos en sus publicaciones.
3	El usuario valora dicho feedback con un 5.
4	El sistema procesa la información y envía la información al servidor.
5	El servidor envía la notificación al usuario.
6	El usuario autor de ese feedback recibe una notificación con un mensaje que indica la nota con la que le han valorado.
Excepción en 3 - Misma nota en la valoración	**Excepción**
1	El usuario valora dicho feedback con la misma nota que anteriormente.
2	El sistema procesa la información y no envía la información al servidor.
3	El usuario autor del feedback no recibe una notificación.
Frecuencia esperada	200 al dia
Importancia	No vital
Prioridad	Largo plazo
Comentarios	N/A

Cuadro A.13: Especificación del caso de uso CU16

Anexo B

Informes QUIS

Questionnaire for User Interface Satisfaction

Aprendiz 1

OVERALL REACTIONS TO THE SOFTWARE			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
1		terrible						x					wonderful	
2		difficult								x			easy	
3		frustrating							x				satisfying	
4		inadequate								x			adequate	
5		dull							x				stimulating	
6		rigid						x					flexible	
SCREEN			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
7	Reading characters on the screen	hard to read								x			easy to read	
8	Highlighting simplifies task	not at all											very much	x
9	Organization of information	confusing								x			very clear	
10	Sequence of screens	confusing								x			very clear	
TERMINOLOGY AND SYSTEM INFORMATION			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
11	Use of terms throughout system	inconsistent									x		consistent	
12	Rerminology related to task	never								x			always	
13	Position of messages on screen	inconsistent									x		consistent	
14	Prompts for input	confusing								x			clear	
15	Computer nforms about what its progress	never								x			always	
16	Error messages	unhelpful										x	helpful	
LEARNING			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
17	Learning to operate the system	difficult									x		easy	
18	Exploring new features by trial and error	difficult									x		easy	
19	Remembering names and use of commands	difficult											easy	x
20	Performing task is straightforward	never									x		always	
21	Help messages on the screen	unhelpful											helpful	x
22	Supplemental reference materials	confusing											clear	x
SYSTEM CAPABILITIES			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
23	System speed	too slow							x				fast enough	
24	System reliability	unreliable								x			reliable	
25	System tends to be	noisy									x		quiet	
26	Correcting your mistakes	difficult							x				easy	
27	Designed for all level of users	never								x			always	

	List the most **negative** aspect(s)		List the most **positive** aspect(s)
1	Design	1	Ammount of information shown in previews
2	Usability	2	Variety of content
3	Distribution of the elements	3	Easy navigation

TEST PERSON INFORMATION

Age:	22
Gender:	Male
Level of education:	Graduated
Hours spent using a computer per week on average:	60
Do you blog?	No
Do you usually read a newspaper?	No
Do you know/use:	
- Facebook:	Yes
- Linkedin:	Yes
- Github:	Yes

Questionnaire for User Interface Satisfaction

Aprendiz 2

OVERALL REACTIONS TO THE SOFTWARE			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
1		terrible										x	wonderful	
2		difficult										x	easy	
3		frustrating									x		satisfying	
4		inadequate										x	adequate	
5		dull								x			stimulating	
6		rigid							x				flexible	
SCREEN			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
7	Reading characters on the screen	hard to read										x	easy to read	
8	Highlighting simplifies task	not at all											very much	x
9	Organization of information	confusing								x			very clear	
10	Sequence of screens	confusing								x			very clear	
TERMINOLOGY AND SYSTEM INFORMATION			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
11	Use of terms throughout system	inconsistent									x		consistent	
12	Terminology related to task	never								x			always	
13	Position of messages on screen	inconsistent									x		consistent	
14	Prompts for input	confusing								x			clear	
15	Computer informs about what its progress	never										x	always	
16	Error messages	unhelpful							x				helpful	
LEARNING			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
17	Learning to operate the system	difficult										x	easy	
18	Exploring new features by trial and error	difficult										x	easy	
19	Remembering names and use of commands	difficult											easy	x
20	Performing task is straightforward	never										x	always	
21	Help messages on the screen	unhelpful											helpful	x
22	Supplemental reference materials	confusing											clear	x
SYSTEM CAPABILITIES			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
23	System speed	too slow										x	fast enough	
24	System reliability	unreliable									x		reliable	
25	System tends to be	noisy										x	quiet	
26	Correcting your mistakes	difficult									x		easy	
27	Designed for all level of users	never										x	always	

	List the most **negative** aspect(s)		List the most **positive** aspect(s)
1	Would like an error message to tell me I cannot edit my profile without clicking the button	1	Easy to navigate through the system
2		2	
3		3	

TEST PERSON INFORMATION

Age:	21
Gender:	Male
Level of education:	Graduate
Hours spent using a computer per week on average:	42
Do you blog?	Yes
Do you usually read a newspaper?	Yes
Do you know/use:	
- Facebook:	Yes
- Linkedin:	Yes
- Github:	Yes

Questionnaire for User Interface Satisfaction

Experto 1

OVERALL REACTIONS TO THE SOFTWARE			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
1		terrible						x					wonderful	
2		difficult							x				easy	
3		frustrating								x			satisfying	
4		inadequate							x				adequate	
5		dull									x		stimulating	
6		rigid								x			flexible	
SCREEN			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
7	Reading characters on the screen	hard to read										x	easy to read	
8	Highlighting simplifies task	not at all											very much	x
9	Organization of information	confusing						x					very clear	
10	Sequence of screens	confusing									x		very clear	
TERMINOLOGY AND SYSTEM INFORMATION			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
11	Use of terms throughout system	inconsistent								x			consistent	
12	Terminology related to task	never										x	always	
13	Position of messages on screen	inconsistent							x				consistent	
14	Prompts for input	confusing									x		clear	
15	Computer nforms about what its progress	never								x			always	
16	Error messages	unhelpful							x				helpful	
LEARNING			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
17	Learning to operate the system	difficult									x		easy	
18	Exploring new features by trial and error	difficult									x		easy	
19	Remembering names and use of commands	difficult											easy	x
20	Performing task is straightforward	never									x		always	
21	Help messages on the screen	unhelpful											helpful	x
22	Supplemental reference materials	confusing											clear	x
SYSTEM CAPABILITIES			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
23	System speed	too slow								x			fast enough	
24	System reliability	unreliable							x				reliable	
25	System tends to be	noisy								x			quiet	
26	Correcting your mistakes	difficult							x				easy	
27	Designed for all level of users	never							x				always	

	List the most **negative** aspect(s)		List the most **positive** aspect(s)
1	An overloaded interface	1	The funcionality
2		2	The interface
3		3	

TEST PERSON INFORMATION

Age:	22
Gender:	Male
Level of education:	Graduate
Hours spent using a computer per week on average:	56 hours
Do you blog?	yes
Do you usually read a newspaper?	No
Do you know/use:	
- Facebook:	Yes
- Linkedin:	No
- Github:	Yes

Questionnaire for User Interface Satisfaction

Experto 2

OVERALL REACTIONS TO THE SOFTWARE			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
1		terrible							x				wonderful	
2		difficult										x	easy	
3		frustrating										x	satisfying	
4		inadequate									x		adequate	
5		dull						x					stimulating	
6		rigid										x	flexible	
SCREEN			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
7	Reading characters on the screen	hard to read										x	easy to read	
8	Highlighting simplifies task	not at all											very much	x
9	Organization of information	confusing										x	very clear	
10	Sequence of screens	confusing										x	very clear	
TERMINOLOGY AND SYSTEM INFORMATION			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
11	Use of terms throughout system	inconsistent									x		consistent	
12	Terminology related to task	never									x		always	
13	Position of messages on screen	inconsistent										x	consistent	
14	Prompts for input	confusing									x		clear	
15	Computer informs about what its progress	never										x	always	
16	Error messages	unhelpful								x			helpful	
LEARNING			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
17	Learning to operate the system	difficult										x	easy	
18	Exploring new features by trial and error	difficult										x	easy	
19	Remembering names and use of commands	difficult											easy	x
20	Performing task is straightforward	never										x	always	
21	Help messages on the screen	unhelpful											helpful	x
22	Supplemental reference materials	confusing											clear	x
SYSTEM CAPABILITIES			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
23	System speed	too slow										x	fast enough	
24	System reliability	unreliable									x		reliable	
25	System tends to be	noisy										x	quiet	
26	Correcting your mistakes	difficult								x			easy	
27	Designed for all level of users	never										x	always	

	List the most **negative** aspect(s)		List the most **positive** aspect(s)
1	Distribution of the elements	1	Easy to use and navigate
2		2	Easy to create elements
3		3	

TEST PERSON INFORMATION

Age:	21
Gender:	Male
Level of education:	Graduate
Hours spent using a computer per week on average:	25
Do you blog?	No
Do you usually read a newspaper?	Yes
Do you know/use:	
- Facebook:	Yes
- Linkedin:	Yes
- Github:	No